Collections, Sets and Types

Gilles Dowek

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET AUTOMATIQUE

Collections, sets and types

Gilles Dowek

N ° 2708

Novembre 1995

PROGRAMME 2

Calcul symbolique,
programmation
et génie logiciel

ISSN 0249-6399

INRIA
ROCQUENCOURT

Collections, sets and types

Gilles Dowek *

Programme 2 — Calcul symbolique, programmation et génie logiciel

Projet Coq

Rapport de recherche n ° 2708 — Novembre 1995 — 33 pages

Abstract: We give a first order formulation of Church's type theory in which types are mere sets. This formulation is obtained by replacing λ-calculus by a language of combinators (skolemized comprehension schemes), introducing a distinction between propositions and their contents, relativizing quantifiers and at last replacing typing predicates by membership to some sets. The theory obtained this way has both a type theoretical flavor and a set theoretical one. Like set theory, it is a first order theory, and it uses only one notion of collection. Like type theory, it gives an explicit notation for objects, a primitive notion of function and propositions are objects.

Key-words: type, set, type theory, set theory

(Résumé : tsvp)

* Gilles.Dowek@inria.fr

Unité de recherche INRIA Rocquencourt

Collections, ensembles et types

Résumé : Nous donnons une formulation au premier ordre de la théorie des types de Church, dans laquelle les types sont des ensembles ordinaires. Cette formulation est obtenue en remplaçant le λ-calcul par un langage de combinateurs (schémas de compréhension skolémisé), en introduisant une distinction entre les propositions et leurs contenus, en relativisant les quantificateurs et enfin en remplaçant les prédicats de typage par l'appartenance à des ensembles. La théorie ainsi obtenue présente à la fois des aspects de théorie des types et des aspects de théorie des ensembles. Comme la théorie des ensembles, c'est une théorie du premier ordre et elle repose sur une unique notion de collection. Comme la théorie des types, elle fournit une notation explicite pour les objets, une notion primitive de fonction et les propositions sont des objets.

Mots-clé : type, ensemble, théorie des types, théorie des ensembles

Introduction

A *set* is a collection of objects defined by a characteristic property of its elements, for instance "the set of natural numbers lower than or equal to 3". At a first glance, a *type* is also a collection of objects defined by a characteristic property of its elements, for instance "the type of natural numbers". A difference between the notions of set and type lies in the fact that a given object usually belongs to only one type while it may belong to several sets. For instance, the type of 0 is *nat* (the type of natural numbers) while 0 belongs to several sets $\{x \mid x \leq 3\}$, $\{x \mid x = 0\}$, $\{x \mid \exists y\ x = 2 * y\}$, *etc.* Also the fact that a given object belongs to a type is often a decidable judgement, while the fact that a given object belongs to a set is usually not. At last, the type of an expression restricts its use. For instance, as 0 has type *nat* and *true* has type *bool*, the proposition $0 \leq 0$ is well-formed and the proposition $true \leq true$ is not.

Thus a postulate underlying the use of (simply) typed languages is that among all the properties verified by an object, a single one, its type, determines the uses of this object. In many situations this postulate seems to be too restrictive. For instance, the expression $\sqrt{a}$ is defined when a is a positive real number and $1/a$ is defined when a is non zero complex number. Thus more liberal type disciplines have been designed (overloading, polymorphism, subtyping, *etc.*) For instance, we may consider the types *complex*, *real*, *non-zero* and *positive*, *non-zero* and *real* being subtypes of *complex* and *positive* being a subtype of *real*. When such extensions are considered, unicity of type is lost and decidability sometimes also. Moreover there seems to be very few differences between the judgement "1 belongs to the type *positive*" and the proposition "$1 \in R^+$". There is a conceptual difficulty to explain why type and set are two distinct notions and it seems that types should be mere sets. This advocates for founding mathematics on a theory with a single notion of collection, such as set theory, rather than one with two distinct notions, such as type theory.

Another reason advocating for set theory are that it is a first order theory while type theory is not. At last, for abstract mathematics set theory is better suited than type theory. If we develop, for instance, results about groups in type theory, we cannot apply these results both to the group of integers and to the group of bijective functions mapping integers to integers, because when we develop these results, we need to give a fixed type to the elements of the group. This problem is usually solved by using "typical ambiguity" i.e. more or less type variables.

However, in some situations, type theory is better suited than set theory. For instance, proofs-as-objects interpretations are always developed in typed languages [8, 34, 21, 22, 12]. Also when one is interested in expressing computations as rewrite rules on the expressions of the language, type theories seem better suited than set theories [34, 21, 22, 12]. At last automated theorem proving methods have been developed for type theory [3, 28, 29] and very rarely for set theory.

Several reasons explain these success of type theory. (1) Type theory provides an explicit notation for objects (λ-calculus) while set theory merely provides axioms expressing the existence of some objects. (2) Functions are primitive objects in type theory and the object obtained by applying a function to some object has a notation, while it has not in set theory as functions are coded as relations. (3) Meaningless statements such as $1 \in 0$ are forbidden by the syntax of type theory but not by the one of set theory. (4) Typing judgements are decidable in type theory (this fact is used in the functional interpretation of proofs, as proofs judgements must be decidable). (5) Propositions are objects in type theory and not in set theory (with the exception of Morse's set theory [39]).

We give in this paper a first order formulation of Church's theory of types where types are mere sets. This theory is obtained by first considering a formulation with combinators (skolemized comprehension schemes) instead of λ-calculus [14], introducing a distinction between propositions and their contents, relativizing quantifiers as in [20, 15, 42] and at last replacing the typing predicates by belongness to some sets.

The theory obtained this way has both a type theoretical and a set theoretical flavor. Like set theory, there is only one notion of collection, the theory is expressed in a first order setting and typical ambiguity is not required. Like type theory, it provides an explicit notation for objects, functions are primitive objects and propositions are objects. The decidability of typing judgements can be seen as the decidability of a fragment of the language. The formation of meaningless propositions is not yet forbidden as this requirement

is incompatible with the expression in a first order setting, we suggest at the end of the paper an extension of first order logic that would allow to forbid the formation of these propositions.

This formulation permits to simplify the proofs of Henkin's completeness theorem and of a variant of Miller's higher order skolemization theorem.

1 Typed first order logic

1.1 Syntax

Definition A typed first order language (see [20] for a detailed presentation) is given by

- a denumerable collection $\mathcal{T}$ of types,
- for each type T of $\mathcal{T}$, a denumerably infinite collection V_T of variables, such that if $T \neq U$ then V_T and V_U are disjoint,
- a denumerable collection of function symbols, to each function symbol f is associated an element of $\mathcal{T}^{n+1}$ $(n \geq 0)$ called its rank,
- a denumerable collection of predicate symbols, to each predicate symbol P is associated an element of $\mathcal{T}^n$ $(n \geq 0)$ called its rank.

Definition *Terms* of type T are inductively defined as

- variables of V_T are terms of type T,
- if f is a function symbol of rank $(T_1, ..., T_n, T_{n+1})$ and t_1, ..., t_n are terms of type $T_1, ..., T_n$ then $(f\ t_1\ ...\ t_n)$ is a term of type T_{n+1}.

Definition *Propositions* is inductively defined as

- if P is a predicate symbol of rank $(T_1, ..., T_n)$ and t_1, ..., t_n are terms of type $T_1, ..., T_n$ then $(P\ t_1\ ...\ t_n)$ is a proposition,
- $\top$, $\bot$ are propositions (resp. *truth* and *falsehood*),
- if A is a proposition then $\neg A$ is a proposition,
- if A and B are propositions then $A \wedge B$, $A \vee B$, $A \Rightarrow B$, $A \Leftrightarrow B$ are propositions,
- if A is a proposition and x a variable then $\forall x\ A$ and $\exists x\ A$ are propositions.

1.2 Proofs

Proof rules for typed first order logic are the same as the ones for first order logic, with an extra side condition for quantifiers rules, that a variable of type T can only be substituted by a term of type T. Natural deduction, sequent calculus and Frege-Hilbert systems can be used. We give below a formulation of natural deduction. As usual substitution is defined in such a way that variable captures are avoided.

Definition (Substitution)

- $x[x \leftarrow b] = b$,
- $y[x \leftarrow b] = y$,

- $(f\ c_1\ ...\ c_n)[x \leftarrow b] = (f\ c_1[x \leftarrow b]\ ...\ c_n[x \leftarrow b])$,
- $(P\ c_1\ ...\ c_n)[x \leftarrow b] = (P\ c_1[x \leftarrow b]\ ...\ c_n[x \leftarrow b])$,
- $\top[x \leftarrow b] = \top$, $\bot[x \leftarrow b] = \bot$,
- $(\neg A)[x \leftarrow b] = \neg(A[x \leftarrow b])$,
- $(A \wedge B)[x \leftarrow b] = (A[x \leftarrow b]) \wedge (B[x \leftarrow b])$, $(A \vee B)[x \leftarrow b] = (A[x \leftarrow b]) \vee (B[x \leftarrow b])$,
  $(A \Rightarrow B)[x \leftarrow b] = (A[x \leftarrow b]) \Rightarrow (B[x \leftarrow b])$, $(A \Leftrightarrow B)[x \leftarrow b] = (A[x \leftarrow b]) \Leftrightarrow (B[x \leftarrow b])$,
- $(\forall y\ A)[x \leftarrow b] = \forall z\ (A[y \leftarrow z][x \leftarrow b])$ where z is a fresh variable, i.e. a variable not occurring in $\forall x\ A$ or b,
- $(\exists y\ A)[x \leftarrow b] = \exists z\ (A[y \leftarrow z][x \leftarrow b])$ where z is a fresh variable, i.e. a variable not occurring in $\exists x\ A$ or b.

Definition (Proof rules)

$$\frac{A \in \Gamma}{\Gamma \vdash A}\ \text{axiom}$$

$$\frac{\Gamma\ A \vdash B}{\Gamma \vdash A \Rightarrow B}\ \Rightarrow\text{-intro}$$

$$\frac{\Gamma \vdash A \Rightarrow B \quad \Gamma \vdash A}{\Gamma \vdash B}\ \Rightarrow\text{-elim}$$

$$\frac{\Gamma\ A \vdash B \quad \Gamma\ B \vdash A}{\Gamma \vdash A \Leftrightarrow B}\ \Leftrightarrow\text{-intro}$$

$$\frac{\Gamma \vdash A \Leftrightarrow B \quad \Gamma \vdash A}{\Gamma \vdash B}\ \Leftrightarrow\text{-elim}$$

$$\frac{\Gamma \vdash A \Leftrightarrow B \quad \Gamma \vdash B}{\Gamma \vdash A}\ \Leftrightarrow\text{-elim}$$

$$\frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \wedge B}\ \wedge\text{-intro}$$

$$\frac{\Gamma \vdash A \wedge B}{\Gamma \vdash A}\ \wedge\text{-elim}$$

$$\frac{\Gamma \vdash A \wedge B}{\Gamma \vdash B}\ \wedge\text{-elim}$$

$$\frac{\Gamma \vdash A}{\Gamma \vdash A \vee B}\ \vee\text{-intro}$$

$$\frac{\Gamma \vdash B}{\Gamma \vdash A \vee B}\ \vee\text{-intro}$$

$$\frac{\Gamma \vdash A \vee B \quad \Gamma\ A \vdash C \quad \Gamma\ B \vdash C}{\Gamma \vdash C}\ \vee\text{-elim}$$

$$\frac{\Gamma\ A \vdash \bot}{\Gamma \vdash \neg A}\ \neg\text{-intro}$$

$$\frac{\Gamma \vdash A \quad \Gamma \vdash \neg A}{\Gamma \vdash \bot}\ \neg\text{-elim}$$

$$\frac{}{\Gamma \vdash \top}\ \top\text{-intro}$$

$$\frac{\Gamma \vdash \bot}{\Gamma \vdash A}\ \bot\text{-elim}$$

$$\frac{\Gamma \vdash A}{\Gamma \vdash \forall x\ A}\ \forall\text{-intro} \quad x \text{ not free in } \Gamma$$

$$\frac{\Gamma \vdash \forall x\ A}{\Gamma \vdash A[x \leftarrow t]}\ \forall\text{-elim} \quad x \text{ and } t \text{ have the same type}$$

$$\frac{\Gamma \vdash A[x \leftarrow t]}{\Gamma \vdash \exists x\ A}\ \exists\text{-intro} \quad x \text{ and } t \text{ have the same type}$$

$$\frac{\Gamma \vdash \exists x\ A \quad \Gamma A \vdash B}{\Gamma \vdash B}\ \exists\text{-elim} \quad x \text{ not free in } B \text{ or in } \Gamma$$

$$\frac{}{\Gamma \vdash A \vee \neg A}\ \text{classic}$$

where Γ is a finite set of propositions and A, B and C are propositions.

1.3 Models

Definition A model $\mathcal{M}$ for typed first order logic based on a collection $\mathcal{T}$ of types is given by a family $(M_T)_{T \in \mathcal{T}}$ of non empty sets, for each function symbol f of type $(T_1, ..., T_n, T_{n+1})$ a function $\hat{f}$ from $M_{T_1} \times ... \times M_{T_n}$ to $M_{T_{n+1}}$, for each predicate symbol P of type $(T_1, ..., T_n)$ a subset $\hat{P}$ of $M_{T_1} \times ... \times M_{T_n}$.

Definition Let $\mathcal{M}$ be a model, an assignment onto $\mathcal{M}$ is function φ associating to each variable x of V_T an element of M_T. If φ is an assignment, x a variable of V_T and a an element of M_T, we write $\varphi + (x, a)$ for the assignment ψ such that $\psi(y) = a$ if $y = x$ and $\psi(y) = \varphi(y)$ otherwise.

Definition Let $\mathcal{M}$ be a model and φ an assignment, we define a function $\overline{\varphi}$ that associates to each term of type T an element of M_T and to each proposition an element of $\{0, 1\}$.

- $\overline{\varphi}(x) = \varphi(x)$,
- $\overline{\varphi}(f\ t_1\ ...\ t_n) = \hat{f}(\overline{\varphi}(t_1), ..., \overline{\varphi}(t_n))$,
- $\overline{\varphi}(P\ t_1\ ...\ t_n) = 1$ if $(\overline{\varphi}(t_1), ..., \overline{\varphi}(t_n))$ is in $\hat{P}$ and 0 otherwise,
- $\overline{\varphi}(\top) = 1$,
- $\overline{\varphi}(\bot) = 0$,
- $\overline{\varphi}(\neg A) = 1$ if $\overline{\varphi}(A) = 0$ and 0 otherwise,
- $\overline{\varphi}(A \wedge B) = 1$ if $\overline{\varphi}(A) = \overline{\varphi}(B) = 1$ and 0 otherwise,
- $\overline{\varphi}(A \vee B) = 1$ if $\overline{\varphi}(A) = 1$ or $\overline{\varphi}(B) = 1$ and 0 otherwise,
- $\overline{\varphi}(A \Rightarrow B) = 1$ if $\overline{\varphi}(A) = 0$ or $\overline{\varphi}(B) = 1$ and 0 otherwise,
- $\overline{\varphi}(A \Leftrightarrow B) = 1$ if $\overline{\varphi}(A) = \overline{\varphi}(B)$ and 0 otherwise,
- $\overline{\varphi}(\forall x\ A) = 1$ if for every a in M_T (where T is the type of x), $\overline{\varphi + (x, a)}(A) = 1$ and 0 otherwise,
- $\overline{\varphi}(\exists x\ A) = 1$ if there exists a in M_T (where T is the type of x) such that $\overline{\varphi + (x, a)}(A) = 1$ and 0 otherwise.

Definition A model $\mathcal{M}$ is a *model of a proposition* A (A is *valid* in $\mathcal{M}$) if $\overline{\varphi}(A) = 1$ for every assignment φ. A model is a model of a set of propositions Γ if it is a model of each proposition of Γ.

Theorem (Completeness) Γ has a model if and only if $\Gamma \not\vdash \bot$ [26, 46, 20].

Corollary $\Gamma \vdash A$ if and only if every model of Γ is a model of A.

2 Set theory and type theory

2.1 Naïve set theory

2.1.1 Functions and predicates

In a first order language, individual symbols denote elements of the domain, predicate symbols denote sets of elements of the domain and function symbols denote functions over the domain.

More expressions denoting elements of the domain can be built as terms. Facts concerning all the elements of the domain or some unspecified elements of the domain can be expressed using variables and quantifiers. But, it is impossible to construct expressions denoting more sets of elements of the domain, or more functions over the domain than the ones given as primitive symbols. It is also impossible to express facts concerning all (or some unspecified) sets of elements of the domain or all (or some unspecified) functions over the domain. For instance, in first order arithmetic we cannot express the proposition

Every non empty set of natural numbers has a greatest lower bound

Any first order theory can be extended with sets and functions. A set is written $\{x_1, ..., x_n \mid P\}$ where $x_1, ..., x_n$ are variables and P is a proposition, in the same way a function is written $x_1, ..., x_n \mapsto t$ where $x_1, ..., x_n$ are variables and t is a term. In this extension, variables may denote arbitrarily elements of the domain, sets of elements of the domain, functions over the domain, and also sets of sets, sets of functions, functions over sets, functions over functions, *etc.* Any term can be understood as denoting an individual, a set or a function of arbitrary arity, thus the notion of arity is lost, also is the distinction between individual symbols, function symbols and predicate symbols which are replaced by a single syntactical category of *primitive symbols*.

Definition *Terms* and *propositions* are inductively defined as

- variables are terms,
- primitive symbols are terms,
- if f is a term and $a_1, ..., a_n$ are terms then $(f\ a_1\ ...\ a_n)$ is a term.
- if P is a proposition and $x_1, ..., x_n$ are variables then $\{x_1, ..., x_n \mid P\}$ is a term,
- if t is a term and $x_1, ..., x_n$ are variables then $x_1, ..., x_n \mapsto t$ is a term.
- If P is a term and $a_1, ..., a_n$ are terms then $(P\ a_1\ ...\ a_n)$ is a proposition,
- $\top$ and $\bot$ are propositions,
- if A is a proposition then $\neg A$ is a proposition,
- if A and B are propositions then $A \wedge B$, $A \vee B$, $A \Rightarrow B$, $A \Leftrightarrow B$ are propositions,
- if A is a proposition and x a variable then $\forall x\ A$ and $\exists x\ A$ are propositions.

Definition The tems of the form $\{x_1, ..., x_n \mid P\}$ and $x_1, ..., x_n \mapsto t$ are called *abstractions*.

Definition (Substitution)

- $x[x \leftarrow b] = b$,
- $y[x \leftarrow b] = y$,
- $(c\ d)[x \leftarrow b] = (c[x \leftarrow b]\ d[x \leftarrow b])$,
- $(y_1, ..., y_n \mapsto c)[x \leftarrow b] = z_1, ..., z_n \mapsto (c[y_1 \leftarrow z_1]...[y_n \leftarrow z_n][x \leftarrow b])$ where $z_1, ..., z_n$ are fresh variables, i.e. variables not occurring in $y_1, ..., y_n \mapsto c$ or b,
- $\{y_1, ..., y_n \mid P\}[x \leftarrow b] = \{z_1, ..., z_n \mid (P[y_1 \leftarrow z_1]...[y_n \leftarrow z_n][x \leftarrow b]\}$ where $z_1, ..., z_n$ are fresh variables, i.e. variables not occurring in $\{y_1, ..., y_n \mid P\}$ or b,
- $\top[x \leftarrow b] = \top$, $\bot[x \leftarrow b] = \bot$,
- $(\neg A)[x \leftarrow b] = \neg(A[x \leftarrow b])$,
- $(A \wedge B)[x \leftarrow b] = (A[x \leftarrow b]) \wedge (B[x \leftarrow b])$, $(A \vee B)[x \leftarrow b] = (A[x \leftarrow b]) \vee (B[x \leftarrow b])$, $(A \Rightarrow B)[x \leftarrow b] = (A[x \leftarrow b]) \Rightarrow (B[x \leftarrow b])$, $(A \Leftrightarrow B)[x \leftarrow b] = (A[x \leftarrow b]) \Leftrightarrow (B[x \leftarrow b])$,
- $(\forall y\ A)[x \leftarrow b] = \forall z\ (A[y \leftarrow z][x \leftarrow b])$ where z is a fresh variable, i.e. a variable not occurring in $\forall x\ A$ or b,
- $(\exists y\ A)[x \leftarrow b] = \exists z\ (A[y \leftarrow z][x \leftarrow b])$ where z is a fresh variable, i.e. a variable not occurring in $\exists x\ A$ or b.

To the axioms already present in the the first order theory, we add axioms expressing the replacement of formal arguments of a set or a function by the actual ones (conversion axioms) and axioms defining the equality of two sets or two functions (extensionality axioms). We take the universal closure of the following propositions.
Conversion:

$$\forall x_1\ ...\ \forall x_n\ (\{x_1, ..., x_n \mid P\}\ x_1\ ...\ x_n) \Leftrightarrow P$$
$$\forall x_1\ ...\ \forall x_n\ ((x_1, ..., x_n \mapsto t)\ x_1\ ...\ x_n) = t$$

Extensionality:

$$\forall f\ \forall g\ (\forall x_1\ ...\ \forall x_n\ (f\ x_1\ ...\ x_n) = (g\ x_1\ ...\ x_n)) \Rightarrow f = g$$
$$\forall P\ \forall Q\ (\forall x_1\ ...\ \forall x_n\ (P\ x_1\ ...\ x_n) \Leftrightarrow (Q\ x_1\ ...\ x_n)) \Rightarrow P = Q$$

Deduction rules are the usual ones.

Definition *Naïve set theory* is the extension of the first order theory of equality with functions and sets.

We do not take any kind of mathematical objects (numbers, *etc.*) as primitive as once we have sets and functions we can construct internally all the mathematical objects we need. Thus we take as axioms the universal closure of the following propositions.
Conversion:

$$\forall x_1\ ...\ \forall x_n\ (\{x_1, ..., x_n \mid P\}\ x_1\ ...\ x_n) \Leftrightarrow P$$
$$\forall x_1\ ...\ \forall x_n\ ((x_1, ..., x_n \mapsto t)\ x_1\ ...\ x_n) = t$$

Extensionality:

$$\forall f\ \forall g\ (\forall x_1\ ...\ \forall x_n\ (f\ x_1\ ...\ x_n) = (g\ x_1\ ...\ x_n)) \Rightarrow f = g$$
$$\forall P\ \forall Q\ (\forall x_1\ ...\ \forall x_n\ (P\ x_1\ ...\ x_n) \Leftrightarrow (Q\ x_1\ ...\ x_n)) \Rightarrow P = Q$$

Equality:

$$\forall x\ (x = x)$$
$$\forall a\ \forall b\ (a = b) \Rightarrow (P[x \leftarrow a] \Rightarrow P[x \leftarrow b])$$

2.1.2 The Comprehension schemes

In the presentation of a theory we can either choose to give notations for objects and axioms expressing the properties of these objects, or to give axioms expressing the existence of objects verifying the desired properties. For instance, relations with a maximal element can either be defined by the language $\leq$, M and the axiom

$$\forall x\ (x \leq M)$$

or by the language $\leq$ and the axiom

$$\exists y\ \forall x\ (x \leq y)$$

From the second formulation we can produce the first by skolemizing the axioms (in this case we skolemize the external existential quantifier of the axiom).

In the same way instead of having an explicit notation for sets and functions, we can merely state axioms expressing their existence (comprehension schemes). We take as axioms the universal closure of the following propositions.

$$\exists A\ \forall x_1\ ...\ \forall x_n\ ((A\ x_1\ ...\ x_n) \Leftrightarrow P)$$

where A is not free in P,

$$\exists f\ \forall x_1\ ...\ \forall x_n\ ((f\ x_1\ ...\ x_n) = t)$$

where f is not free in t.

If we skolemize these axioms we get primitive symbols that roughly look like the terms $x_1, ..., x_n \mapsto t$ and $\{x_1, ..., x_n \mid P\}$. The comprehension schemes roughly become

$$\forall x_1\ ...\ \forall x_n\ ((\{x_1, ..., x_n \mid P\}\ x_1\ ...\ x_n) \Leftrightarrow P)$$

$$\forall x_1\ ...\ \forall x_n\ (((x_1, ..., x_n \mapsto t)\ x_1\ ...\ x_n) = t)$$

which are more or less the conversion axioms [6].

There are however some differences between the theory obtained by skolemizing the comprehension schemes and the theory based on the full notation $\mapsto$, $\{\ |\ \}$.

First, as abstractions are primitive symbols in the skolemized language, there is no substitution under abstractions. Then, when we skolemize the comprehension schemes, we get symbols $x_1, ..., x_n \mapsto t$ and $\{x_1, ..., x_n \mid P\}$ only for terms t and propositions P that do not contain further abstractions, i.e. there is no nested abstractions. So the equivalence of the these two formulations is not a mere consequence of Skolem's theorem. For naïve set theory the equivalence of these formulations is trivial (since, as we shall see, both formulations are inconsistent). For *set theory* and *type theory*, studied below, the equivalence is proved in [14] (slight adaptation of the proof of [14] is needed for type theory as formulated below).

Skolemizing the comprehension schemes gives a presentation of the theory with an explicit notation for objects, but this notation is closer to a language of combinators than to the full notation $\mapsto$, $\{\ |\ \}$.

2.1.3 Flattening

In a first order language, in a term of the form $(f\ a_1\ ...\ a_n)$, f is a function symbol, thus this term can be defined as a tree whose root is labeled by the symbol f and whose sons are $a_1, ..., a_n$. In naïve set theory, we cannot look at the term $(f\ a_1\ ...\ a_n)$ as a tree whose root is labeled by f, as f is also a term. So we define this term as a tree whose root is labeled by a symbol α_n and whose sons are $f, a_1, ..., a_n$. Such a term is better written $(\alpha_n\ f\ a_1\ ...\ a_n)$. In the same way, a proposition $(P\ a_1\ ...\ a_n)$ is better written $(\in_n\ P\ a_1\ ...\ a_n)$.

This way, primitive symbols (i.e. individual symbols, function symbols and predicate symbols of the initial language) are individual symbols (i.e. zero-ary function symbols) in this new language, while the only (non zero-ary) function symbols and predicate symbols are α_n and $\in_n$.

Remark With the comprehension schemes (or the skolmemized comprehension schemes) and the symbols α_n and $\in_n$, naïve set theory is formulated in a first order language.

But this first-order language is very different from the first-order language we started with. If we started, for instance, with first order arithmetic, the symbols 0, S and $\leq$ were respectively an individual symbol, a unary function symbol and a binary predicate symbol. They are now all individual symbols. In first order arithmetics, the domain of the language contains only natural numbers, now it contains all together natural numbers, sets, functions, functions of functions, *etc.*

There are two ways to extend a first order theory. Either we keep the same universe of the discourse and we extend the logic to speak about sets of elements of this universe and functions over this universe, or we extend the universe of the discourse including sets and functions and we keep the same logic. Flattening permits to shift from the first choice to the second.

2.1.4 Currying

Instead of having a notion of function of n variables, we can have only functions of one variable and define a function of n variables f as a function mapping an object a to the function of $n-1$ variables $x_2, ..., x_n \mapsto (f\ a\ x_2\ ...\ x_n)$.

The term $(\alpha_n\ f\ a_1\ ...\ a_n)$ is now written $(\alpha_1\ ...\ (\alpha_1\ f\ a_1)\ ...\ a_n)$ and if we have the notation $\mapsto$, the term $x_1, ..., x_n \mapsto t$ is written $x_1 \mapsto (...\ x_n \mapsto t)$. (The language obtained this way is called λ-calculus.) In the same way a predicate (set) of n variables can be defined as a function mapping an object a to the predicate of $n-1$ variables $\{x_2, ..., x_n \mid (P\ a\ x_2\ ...\ x_n)\}$. The proposition $(\in_n\ A\ a_1\ ...\ a_n)$ is now written $\in_0\ (\alpha_1\ ...\ (\alpha_1\ A\ a_1)\ ...\ a_n)$ and if we have the notation $\{\ \mid\ \}$, the term $\{x_1, ..., x_n \mid P\}$ is then written $x_1 \mapsto ... \mapsto x_n \mapsto \{P\}$. Notice that P is a proposition and $\{P\}$ is a term. The object $\{P\}$ is the content (*lexis*) of the proposition P. In contrast, when we consider the comprehension schemes or the skolemized comprehension schemes, we cannot restrict ourselves to unary schemes that are strictly weaker than the n-ary ones [14].

Let us write α for α_1 and ε for $\in_0$. The axioms are then rephrased as the universal closure of the following propositions.

Comprehension:

$$\exists A\ \forall x_1\ ...\ \forall x_n\ ((\varepsilon\ (\alpha\ ...\ (\alpha\ A\ x_1)\ ...\ x_n)) \Leftrightarrow P)$$

where A is not free in P,

$$\exists f\ \forall x_1\ ...\ \forall x_n\ ((\alpha\ ...\ (\alpha\ f\ x_1)\ ...\ x_n) = t)$$

where f is not free in t.

Extensionality:

$$\forall P\ \forall Q\ ((\varepsilon\ P) \Leftrightarrow (\varepsilon\ Q)) \Rightarrow (P = Q)$$

$$\forall f\ \forall g\ (\forall x\ (\alpha\ f\ x) = (\alpha\ g\ x)) \Rightarrow (f = g)$$

Equality:

$$\forall x\ (x = x)$$

$$\forall a\ \forall b\ (a = b) \Rightarrow (P[x \leftarrow a] \Rightarrow P[x \leftarrow b])$$

where $(a = b)$ is an abbreviation for $(\varepsilon\ (\alpha\ (\alpha\ =\ a)\ b))$.

Remark We could go one step further and identify the proposition $(\varepsilon\ a)$ with the term a and the term $\{P\}$ with the proposition P.

2.1.5 Russell's paradox

Naïve set theory is inconsistent as we can derive the proposition

$$(\varepsilon\ (\{x \mid \neg(x\ x)\}\{x \mid \neg(x\ x)\})) \Leftrightarrow \neg(\varepsilon\ (\{x \mid \neg(x\ x)\}\{x \mid \neg(x\ x)\}))$$

and thus the proposition $\bot$.

The mechanisms of naïve set theory can be presented as follows

propositions

abstraction $\downarrow\uparrow$ application

objects

The abstraction mechanism permits to form objects from propositions (for instance, from the proposition $x \leq 3$, we form the object $\{x \mid x \leq 3\}$), and the application mechanism permit to form propositions using objects (for instance, from $\{x \mid x \leq 3\}$ and 2, we form $(\varepsilon\ (\alpha\ \{x \mid x \leq 3\}\ 2))$). Thus to weaken naïve set theory and make it consistent, one can either weaken the downarrow (and get *set theory*) or weaken the uparrow (and get *type theory*).

2.2 Set theory

In set theory we consider only sets and sets have only one argument. The proposition $(\in_1\ P\ a)$ is written $a \in P$. Equality is a predicate symbol (not a primitive symbol), thus there are no primitive symbols. The definitions above are restated as follows.

Definition *Terms* are variables.

Definition *Propositions* are inductively defined as

- if a and b are terms then $a = b$ is a proposition,
- if a and b are terms then $a \in b$ is a proposition,
- $\top$ and $\bot$ are propositions,
- if A is a proposition then $\neg A$ is a proposition,
- if A and B are propositions then $A \wedge B$, $A \vee B$, $A \Rightarrow B$, $A \Leftrightarrow B$ are propositions,
- if A is a proposition and x a variable then $\forall x\ A$ and $\exists x\ A$ are propositions.

The comprehension scheme

$$\exists X\ \forall a\ (a \in X \Leftrightarrow P)$$

is still powerful enough to express Russell's paradox, so we restrict it to four instances: the subset scheme (or restricted comprehension), the power set axiom, the union axiom and the pairing axiom. We keep the extensionality and equality axioms.

Definition The axioms of set theory are the universal closure of the following propositions.
Subset:

$$\forall y\ \exists X\ \forall z\ ((z \in X) \Leftrightarrow ((z \in y) \wedge P))$$

where X and y are not free in P.
Power set:

$$\forall x\ \exists X\ \forall y\ ((y \in X) \Leftrightarrow \forall z\ ((z \in y) \Rightarrow (z \in x)))$$

Union:

$$\forall x\ \exists X\ \forall y\ ((y \in X) \Leftrightarrow \exists z\ ((y \in z) \wedge (z \in x)))$$

Pairing:

$$\forall x\ \forall y\ \exists X\ \forall z\ ((z \in X) \Leftrightarrow ((z = x) \vee (z = y)))$$

Extensionality:

$$\forall P\ \forall Q\ (\forall x\ (x \in P) \Leftrightarrow (x \in Q)) \Rightarrow (P = Q)$$

Equality:

$$\forall x\ (x = x)$$

$$\forall a\ \forall b\ (a = b) \Rightarrow (P[x \leftarrow a] \Rightarrow P[x \leftarrow b])$$

Remark Set theory is a first order theory.

Remark These axioms are Zermelo's axioms, they can be extended to Zermelo-Fraenkel by adding the replacement scheme. Other extensions are the Von Neuman-Gödel-Bernays set theory and Morse's set theory.

2.3 Type Theory

The other way to weaken naïve set theory is to weaken the rule that permits to form the term $(\alpha\ a\ b)$ whatever the terms a and b are. Thus we add types to naïve set theory and get a typed first order theory.

2.3.1 Higher order languages and type theory

Definition *Types* are inductively defined as

- ι is a type,
- o is a type,
- if T and U are types then $T \to U$ is a type.

Definition A *higher order language* is a typed first order language based on this set of types and containing function symbols $\alpha_{T,U}$ of rank $(T \to U, T, U)$, a predicate symbol ε of rank (o) and some individual symbols of rank (T) for some type T. These individual symbols are called *primitive symbols*.

Thus, terms of type T are thus inductively defined as

- variables of type T are terms of type T,
- primitive symbols of type T are terms of type T,
- if f is a term of type $T \to U$ and a is a term of type T then $(\alpha_{T,U}\ f\ a)$ is a term of type U.

In the same way, propositions are inductively defined as

- if t is a term of type o then $(\varepsilon\ t)$ is a proposition,
- $\top$ and $\bot$ are propositions,
- if A is a proposition then $\neg A$ is a proposition,
- if A and B are propositions then $A \wedge B$, $A \vee B$, $A \Rightarrow B$ and $A \Leftrightarrow B$ are propositions,
- if A is a proposition and x a variable of type T then $\forall x\ A$ and $\exists x\ A$ are propositions.

Definition (Type theory)

Type theory is a theory in the higher order language containing the primitive symbols $=_T$ of type $T \to T \to o$. The axioms are the universal closure of the following propositions.

Comprehension:

$$\exists A\ \forall x_1\ \ldots\ x_n\ (\varepsilon\ (\alpha_{T_n,o}\ \ldots\ (\alpha_{T_1,T_2\to\ldots\to T_n\to o}\ A\ x_1)\ \ldots\ x_n)) \Leftrightarrow P)$$

$$\exists f\ \forall x_1\ \ldots\ x_n\ (\alpha_{T_n,U}\ \ldots\ (\alpha_{T_1,T_2\to\ldots\to T_n\to U}\ f\ x_1)\ \ldots\ x_n)) =_U t$$

Extensionality:

$$\forall P\ \forall Q\ ((\varepsilon\ P) \Leftrightarrow (\varepsilon\ Q)) \Rightarrow (P =_o Q)$$

$$\forall f\ \forall g\ (\forall x\ (\alpha_{T,U}\ f\ x) =_U (\alpha_{T,U}\ g\ x)) \Rightarrow (f =_{T\to U} g)$$

Equality:

$$\forall x\ (x =_T x)$$

$$\forall a\ \forall b\ (a =_T b) \Rightarrow (P[x \leftarrow a] \Rightarrow P[x \leftarrow b])$$

where $(a =_T b)$ is an abbreviation for $(\varepsilon\ (\alpha_{T,o}\ (\alpha_{T,T\to o}\ =_T\ a)\ b))$.

Remark In the comprehension schemes, we can either consider that all the variables free in P (resp. t) are among $x_1, ..., x_n$ (closed schemes) or not (open schemes). In the second case, we take the universal closure of the instances of the schemes to have closed axioms. As shown in [14], these formulations are equivalent. From now on, we consider only the instances of the schemes such that all the variables free in P (resp. t) are among $x_1, ..., x_n$.

Remark The definition of type theory comes in two steps. First we define the notion of higher order language that can be compared to the notion of first order language. Then we define type theory that is a set of axioms expressed in a higher order language and that one can compare to any first order theory, such as arithmetic.

A higher order language permits to introduce set variables and function variables and to quantify over these variables. It does not assert which sets and which functions we are talking about. One could for instance consider a theory such that every function is constant. In type theory the comprehension schemes state the existence of some sets and functions. To have more sets and functions, the descriptions axioms or the axiom of choice are usually added to these axioms [6].

Remark (Avoiding axiom schemes)

In set theory comprehension schemes are indexed by propositions. In type theory, they are indexed by propositions (or terms), and also by types.

It is possible to avoid this indexing by propositions and terms. In set theory, this is done in the Von Neuman-Gödel-Bernays set theory. In type theory, the situation is simpler because, for each type, the comprehension schemes are equivalent to a finite number of their instances.

$$\exists k\ \forall x\ \forall y\ (k\ x\ y) = x$$

$$\exists s\ \forall x\ \forall y\ \forall z\ (s\ x\ y\ z) = (x\ z\ (y\ z))$$

Indeed the combinators K and S are sufficient to express all the λ-terms and all the logical constants can be expressed from equality [25, 2].

The simplicity of this expression (compared to set theory) is due to the fact that propositions are objects in type theory, while they are not in set theory.

2.3.2 Models of type theory and general models

A higher order language is a typed first order language, with function symbols $\alpha_{T,U}$, a predicate symbol ε and individual symbols (primitive symbols). Thus, a model of a higher order language is given by

- a family M_T of non empty sets,
- a family of functions $A_{T,U}$ from $M_{T\to U} \times M_T$ to M_U, denotations of $\alpha_{T,U}$,
- a subset Y of M_o, denotation of ε,
- for each primitive symbol of type T an element of M_T, (for instance, for the language of type theory, an element E_T of $M_{T\to T\to o}$ for every type T, denotation of $=_T$).

Proposition (Completeness) A proposition A is derivable in a theory Γ, if and only if it is valid in every model of Γ.
Proof As a consequence of the completeness theorem for typed first order languages.

Corollary 1 A proposition A is derivable in type theory, if and only if it is valid in every model of type theory.

Corollary 2 (Completeness of type theory)
A proposition A is derivable in type theory from a set of axioms Δ, if and only if it is valid in every model of type theory that is a model of Δ.

Remark Let Γ be a theory and C be the class of models of Γ. We have $\Gamma \vdash A$ if and only if A is valid in all the models of C. Subclasses C' of C may share this property with C. For instance, in first order logic with equality, provable propositions are characterized either by the class of all models of the equality axioms, but also by the class of equality models i.e. models where equality denotes equality in the model. We exhibit now a class of models (the class of *general models*) that is smaller than the class of all models of type theory, but is still such that a proposition is derivable in type theory if and only if it is valid in every general model.

Definition A model is an *equality model* if $(A_{T,o}(A_{T\to o,o}(E_T, a), b)$ is in $Y)$ if and only if $a = b$ in the model.

Proposition Equality models are models of the equality axioms.

Remark There are models of the axiom of equality that are not equality models [5].

Proposition For every model $\mathcal{M}$ of the equality axioms there exists an equality model $\mathcal{M}'$ such that the propositions valid in $\mathcal{M}$ and in $\mathcal{M}'$ are the same.
Proof We let $M'_T = M_T/\equiv$ where $a \equiv b$ if and only if $A_{T,o}(A_{T,T\to o}(E_T, a), b)$ is in Y. If a is an element of M_T we write $\tilde{a}$ for the class of a in M'_T.

The functions $A_{T,U}$ are compatible with the relation $\equiv$ because the proposition

$$\forall f\ \forall g\ \forall x\ \forall y\ ((f = g) \wedge (x = y)) \Rightarrow ((\alpha_{T,U}\ f\ x) = (\alpha_{T,U}\ g\ y))$$

is a consequence of the equality axioms. Thus we can define the denotation of $\alpha_{T,U}$ as $\tilde{A_{T,U}}$ the function mapping $\tilde{a}$ and $\tilde{b}$ to $\tilde{A_{T,U}}(a, b)$.

If a and b are two elements of M_o and $a \equiv b$, then a and b are either both in Y or both not in Y because

$$\forall x\ \forall y\ (x = y) \Rightarrow ((\varepsilon\ x) \Leftrightarrow (\varepsilon\ y))$$

is a consequence of the equality axioms. Thus we can define the denotation of ε as the set $\tilde{Y}$ containing the elements of M'_o containing elements of Y.

By induction over terms structure, the denotation of any term in M'_T is the class of its denotation in M_T and, by induction over propositions structure, the denotation of a proposition in M'_T is its denotation in M_T.

In the following we study only equality models of the axioms of type theory.

Proposition Let $\mathcal{M}$ be an equality model. The model model $\mathcal{M}$ is a model of the axioms

$$\forall x\ \forall y\ (\varepsilon(x) \Leftrightarrow \varepsilon(y)) \Rightarrow (x = y)$$

$$\exists a\ \varepsilon(a) \Leftrightarrow \top$$

$$\exists b\ \varepsilon(b) \Leftrightarrow \bot$$

if and only if M_o has two elements and Y as one element.

Proof The model $\mathcal{M}$ is a model of the axiom

$$\forall x\ \forall y\ (\varepsilon(x) \Leftrightarrow \varepsilon(y)) \Rightarrow (x = y)$$

if and only if for every a and b in M_o,

$$\text{if } a \text{ and } b \text{ are both in } Y \text{ or both not in } Y \text{ then } A_{T,o}(A_{T,T\to o}(E_T, a), b) \text{ is in } Y$$

As this model is a equality model then this condition can be rephrased

$$\text{if } a \text{ and } b \text{ are both in } Y \text{ or both not in } Y \text{ then } a = b$$

In other words

$$Y \text{ has at most one element and } M_o - Y \text{ has at most one element}$$

A model $\mathcal{M}$ is a model of the axioms

$$\exists a\ \varepsilon(a) \Leftrightarrow \top$$

$$\exists b\ \varepsilon(b) \Leftrightarrow \bot$$

if and only if Y has at least one element and $M_o - Y$ also.

Thus $\mathcal{M}$ is a model of these axioms if and only if M_o has two elements and Y as one element.

Definition A a model is *functional* if the elements of $M_{T\to U}$ are functions from M_T to M_U and $A_{T,U}(f, g) = f(g)$.

Proposition Equality functional models are models of the extensionality axiom

$$\forall f\ \forall g\ (\forall x\ (\alpha\ f\ x) = (\alpha\ g\ x)) \Rightarrow (f = g)$$

Proof Let $\mathcal{M}$ be a equality functional model, we want to show that $\mathcal{M}$ is a model of the extensionality axiom i.e. for every a and a' in $M_{T\to U}$

$$\text{if (for every } b \text{ in } M_T,\ A_{U,o}(A_{U,U\to o}(E_U, A_{T,U}(a, b)), A_{T,U}(a', b)) \text{ is in } Y)$$
$$\text{then } A_{T\to U,o}(A_{T\to U,(T\to U)\to o}(E_{T\to U}, a), a') \text{ is in } Y.$$

As $\mathcal{M}$ is an equality model this condition can be rephrased

$$\text{if (for every } b \text{ in } M_T,\ A_{T,U}(a, b) = A_{T,U}(a', b)) \text{ then } a = a'$$

As $\mathcal{M}$ is a functional model this condition can be rephrased

$$\text{if (for every } b \text{ in } M_T,\ a(b) = a'(b)) \text{ then } a = a'$$

which is true as a and a' are functions from M_T to M_U.

Remark There are equality models of the extensionality axiom that are not functional.

Proposition For every equality model $\mathcal{M}$ of the extensionality axiom there exists an equality functional model $\mathcal{M}'$ such that the propositions valid in $\mathcal{M}$ and in $\mathcal{M}'$ are the same.

Proof By induction over type structure we construct a set M'_T and a bijection Φ_T from M_T to M'_T.

We let $M'_\iota = M_\iota$, $\Phi_\iota(x) = x$, $M'_o = M_o$, $\Phi_o(x) = x$. Then we assume that M'_T, M'_U, Φ_T and Φ_U are defined and we define $M'_{T\to U}$ and $\Phi_{T\to U}$. We define the function Ψ from $M_{T\to U}$ to the set of functions from M'_T to M'_U as

$$\Psi(a)(b) = \Phi_U(A_{T,U}(a, \Phi_T^{-1}(b))$$

Ψ is injective as if $\Psi(a) = \Psi(a')$
then for every b in M'_T, $\Psi(a)(b) = \Psi(a')(b)$

thus for every b in M'_T, $\Phi_U(A_{T,U}(a, \Phi_T^{-1}(b)) = \Phi_U(A_{T,U}(a', \Phi_T^{-1}(b)))$
thus for every c in M_T, $\Phi_U(A_{T,U}(a, \Phi_T^{-1}(\Phi_T(c))) = \Phi_U(A_{T,U}(a', \Phi_T^{-1}(\Phi_T(c))))$
i.e. for every c in M_T, $\Phi_U(A_{T,U}(a, c)) = \Phi_U(A_{T,U}(a', c))$
then, as Φ_U is injective, for every c in M_T, $(A_{T,U}(a, c) = (A_{T,U}(a', c))$,
as $\mathcal{M}$ is an equality model of the extensionality axiom we get $a = a'$.
We let $M'_{T\to U} = \Psi(M_{T\to U})$ and $\Phi_{T\to U} = \Psi$.

Then we let $Y' = Y$, $E'_T(a, b) = 1$ if $a = b$ and $E'_T(a, b) = 0$ otherwise, and $A'_{T,U}(a, b) = a(b)$. We prove by induction over term structure that for every term a we have $\overline{\Phi \circ \varphi}(a) = \Phi(\overline{\varphi}(a))$. Then we prove by induction over proposition structure that for every proposition, P we have $\overline{\Phi \circ \varphi}(P) = \overline{\varphi}(P)$.

Thus P is valid in $\mathcal{M}'$ if and only if it is valid in $\mathcal{M}$.

Definition A model is *closed by explicit definitions* if it is a model of the comprehension schemes.

Definition A model is a *general model* ([24]) if

- it is an equality model,
- M_o has two elements and Y has one,
- $M_{T\to U}$ contains functions from M_T to M_U,
- it is closed by explicit definitions.

Proposition For every model $\mathcal{M}$ of type theory there exists a general model $\mathcal{M}'$ such that the propositions valid in $\mathcal{M}$ and in $\mathcal{M}'$ are the same.

Corollary 1 A proposition is valid in every general model if and only if it is provable in type theory.

Corollary 2 (Completeness of type theory)
A proposition A is derivable in type theory from a set of axioms Δ, if and only if it is valid in every general model that is a model of Δ.

Remark There are models of type theory that are not general models. First only equality models (and not all the models of the equality axioms) are general models. Then only functional models (and not all the models of the extensionality axiom) are general models.

Remark The first condition (the model is an equality model) is forgotten in [24] and is added in [5]. This condition is important because if we do not take it, the other conditions do not imply the validity of the axioms. In particular, a non equality functional model need not be a model of the extensionality axiom [5].

Remark The proof of Henkin's completeness theorem above is slightly simpler than the usual one [24, 6]. First, we do not need to re-do the work of Gödel's completeness theorem, but we use this theorem instead. Then, the fact that every equality model of the extensionality axiom can be turned into an equality functional model is a lemma here and its proof is usually mixed with the proof of the completeness theorem.

Remark In Henkin's definition [24], the condition that a general model needs to be closed by explicit definitions is stated as the fact that every λ-term has a denotation. This condition is a quite strong requirement [27] as the notion of *general model* is dependent of the considered language which is undesirable. Instead of taking a presentation of type theory based on λ-calculus we can take one based on the comprehension schemes. Then this condition can be rephrased as the fact that the model is a model of the comprehension schemes [32, 13, 7, 15]. This condition is still dependent of the language as an axiom scheme is a set of axiom indexed by expressions of the language. But now, the same critic to Henkin's models holds for models of (for instance) Peano's arithmetic, as this theory is also expressed using an axiom scheme, and thus a model of this theory cannot be defined independently of the language. Another equivalent condition stated in [4, 5]

using the fact that, for each type, the comprehension schemes are equivalent to a finite number of their instances. This condition is independent of the language.

Remark In Henkin's definition [24], the fact that the elements of $M_{T\to U}$ are functions from M_T to M_U is part of the definition of the notion of model. Thus the extensionality axiom is in some sense built in the definition of models. The idea of taking arbitrary elements in $M_{T\to U}$ (and thus functions for $A_{T,U}$ that are not mere function application) comes from [15, 19]. Also the fact that M_o has two elements is part of this definition and thus the axiom

$$\forall x\ \forall y\ (\varepsilon(x) \Leftrightarrow \varepsilon(y)) \Rightarrow (x = y)$$

is also built in this definition.

Thus, the usual definition of models mixes up two different questions: (1) What is a model of a higher order language ? (2) What is a model of type theory ? In first order model theory, we are used to distinguish between a model of a first order language and a model of a specific theory. These two questions get different answers here. A model of a higher order language is given by a family of sets M_T, a family of function $A_{T,U}$, a subset Y of M_o and denotations for the primitive symbols of the language. A model of type theory is a model of its axioms.

Remark For a presentation of type theory based on the skolemized comprehension schemes (combinators), we have the same definition as for the presentation with the non-skolemized comprehensions schemes, but we need to take in the definition of the model a denotation for each Skolem's symbol and the condition that the model must be a model of the skolemized comprehension schemes instead of the non-skolemized ones. General models can be defined as above.

For a presentation of type theory based on λ-calculus, we can use the translation of the theory based on λ-calculus to the one based on the skolemized comprehension schemes [14] and define the denotation of a term as the denotation of its translation. Then have the equivalence: P is provable in the theory based on λ-calculus if and only if P', the translation of P, is provable in the theory based on the skolemized comprehension schemes, if and only if P' is valid in every model of type theory, if and only if P is valid in every model of type theory.

In a general model, the denotation of λ-terms and propositions can be defined directly by

- $\overline{\varphi}(x) = \varphi(x)$,
- if u is a term of type $T \to U$ and v of type T then $\overline{\varphi}(u\ v) = \overline{\varphi}(u)(\overline{\varphi}(v))$,
- if u is a term of type T then $\overline{\varphi}(x \mapsto u)$ is the function f of $M_{T\to U}$ such that for all $d \in M_T$, $f(d) = \overline{\varphi + (x, d)}(u)$, this function is in $M_{T\to U}$.
- if P is a proposition then $\overline{\varphi}(\{P\})$ is the unique element of Y if $\overline{\varphi}(P) = 1$ and the unique element of $M_o - Y$ otherwise.
- $\overline{\varphi}(\varepsilon\ t) = 1$ if $\overline{\varphi}(t)$ is in Y and 0 otherwise.
- $\overline{\varphi}(\top) = 1$,
- $\overline{\varphi}(\bot) = 0$,
- $\overline{\varphi}(\neg A) = 1$ if $\overline{\varphi}(A) = 0$ and 0 otherwise,
- $\overline{\varphi}(A \wedge B) = 1$ if $\overline{\varphi}(A) = \overline{\varphi}(B) = 1$ and 0 otherwise,
- $\overline{\varphi}(A \vee B) = 1$ if $\overline{\varphi}(A) = 1$ or $\overline{\varphi}(B) = 1$ and 0 otherwise,
- $\overline{\varphi}(A \Rightarrow B) = 1$ if $\overline{\varphi}(A) = 0$ or $\overline{\varphi}(B) = 1$ and 0 otherwise,
- $\overline{\varphi}(A \Leftrightarrow B) = 1$ if $\overline{\varphi}(A) = \overline{\varphi}(B)$ and 0 otherwise,

- $\overline{\varphi}(\forall x\ A) = 1$ if for every a in M_T (where T is the type of x), $\overline{\varphi + (x,a)}(A) = 1$ and 0 otherwise,
- $\overline{\varphi}(\exists x\ A) = 1$ if there exists a in M_T (where T is the type of x) such that $\overline{\varphi + (x,a)}(A) = 1$ and 0 otherwise.

Remark The notion of model of a higher order language can be used also to study stronger theories (with the descriptions axiom, the axiom of choice, the axiom of infinity, axioms of higher order arithmetic, *etc.*) and weaker theories (dropping extensionality, multiple truth values, *etc.*). For instance to show the independence of the extensionality axiom in type theory, Andrews [5] constructs a model in which all axioms of type theory but this one are valid. This model is functional (as this condition is part of the notion of model of [5]) and thus it needs to be a non equality model (as equality functional models are model of the extensionality axiom). A slightly simpler proof can be exhibited by considering a model that is not a functional model as follows.

$$M_o = \{0,1\}$$
$$M_\iota = \{a\}$$
$$M_{\iota\to\iota} = \{f,g\}$$
$$M_{T\to U} = M_U^{M_T} \text{ if } T \neq \iota \text{ or } U \neq \iota$$
$$A_{\iota,\iota}(f,a) = a$$
$$A_{\iota,\iota}(g,a) = a$$
$$A_{T,U}(x,y) = x(y) \text{ if } T \neq \iota \text{ or } U \neq \iota$$
$$Y = \{1\}$$
$$E_T \text{ is the equality on } M_T$$

This result is however slightly different from the one of [5] as we show here the independence of the extensionality axiom from a presentation of type theory with comprehension schemes while the result of [5] shows the independence of the extensionality axiom from type theory with an explicit notation for functions based on λ-calculus, and the equivalence of the two formulations of type theory seems to require the extensionality axiom [14].

2.3.3 Higher-order skolemization (Miller's theorem)

In first order logic, Skolem's theorem expresses the fact that if we replace in a theory an axiom of the form

$$\forall x_1\ ...\ \forall x_n\ \exists y\ P$$

by the proposition

$$\forall x_1\ ...\ \forall x_n\ P[y \leftarrow (f\ x_1\ ...\ x_n)]$$

where f is a new function symbol, i.e. a function symbol that does not occur in the theory, we get a conservative extension of the theory, i.e. the propositions with no occurrence of f are provable in one theory if and only if they are provable in the other.

If we try to generalize this theorem to type theory, letting f be a primitive symbol of type $T_1 \to ... \to T_n \to U$, where T_i is the type of x_i and U the type of y, then the theorem is false. Indeed, the axiom of choice is not provable in type theory [4], but its skolemized form is.

In [36, 37] Miller proposes a Skolem-like theorem for type theory: the symbol f is a *Skolem's symbol* of arity n. Whenever a Skolem's symbol f of arity n occurs in a term it must occur in a subterm of the form $(f\ a_1\ ...\ a_n)$ and the free variables of the a_i's cannot be bound higher in the term. In a presentation of type theory, based on combinators, there are no bound variables and thus Miller's condition simplifies to the fact that f of arity n must occur only in subterms of the form $(f\ a_1\ ...\ a_n)$.

In the first order presentation of type theory with combinators (obtained by skolemizing the comprehension schemes [14]) we can deduce this result from Skolem's theorem. Indeed, if we skolemize an axiom

$$\forall x_1 \ ... \ \forall x_n \ \exists y \ P$$

we introduce a function symbol f of rank $(T_1, ..., T_n, U)$, and not a primitive symbol of type $T_1 \to ... \to T_n \to U$. Thus the symbol f alone is not a term, but it must be applied to some terms $a_1, ..., a_n$ to give a term.

3 Type theory as an untyped first order theory

3.1 Coding typed first order theories into untyped ones

Let us consider a typed first order language. We construct an ordinary first order language as follows [20, 15]:

- to each function symbol f we associate a function symbol f' with the same arity,
- to each predicate symbol P we associate a predicate symbol P' with the same arity,
- to each type T we associate a unary predicate symbol $\mathcal{T}_T$.

We translate terms and propositions as follows.

- $x' = x$,
- $(f \ t_1 \ ... \ t_n)' = (f' \ t'_1 \ ... \ t'_n)$,
- $(P \ t_1 \ ... \ t_n)' = (P' \ t'_1 \ ... \ t'_n)$,
- $\top' = \top$,
- $\bot' = \bot$,
- $(\neg A)' = \neg A'$,
- $(A \wedge B)' = A' \wedge B'$,
- $(A \vee B)' = A' \vee B'$,
- $(A \Rightarrow B)' = A' \Rightarrow B'$,
- $(A \Leftrightarrow B)' = A' \Leftrightarrow B'$,
- $(\forall x \ A)' = \forall x \ (\mathcal{T}_T(x) \Rightarrow A')$ where T is the type of x,
- $(\exists x \ A)' = \exists x \ (\mathcal{T}_T(x) \wedge A')$ where T is the type of x.

We translate a theory by translating each axiom, and adding the following axioms.

$$\exists x \ (\mathcal{T}_T \ x)$$

$$\forall x_1 \ ... \ \forall x_n \ (((\mathcal{T}_{T_1} \ x_1) \wedge ... \wedge (\mathcal{T}_{T_n} \ x_n)) \Rightarrow (\mathcal{T}_{T_{n+1}} \ (f' x_1 \ ... \ x_n)))$$

if f is a function symbol of rank $(T_1, ..., T_n, T_{n+1})$.

Proposition Let Γ be a typed first order theory and Γ' its translation (i.e. the translation of its axioms plus the axioms above). The theory Γ has a model if and only if Γ' has a model.
Proof See [15].

Corollary $\Gamma \vdash P$ if and only if $\Gamma' \vdash P'$.

3.2 Translation of the axioms of type theory

3.2.1 Translation of the axioms of type theory

Comprehension:

$$\exists A\ (\mathcal{T}_{T_1\to\ldots\to T_n\to o}\ A)\wedge\forall x_1\ \ldots\ \forall x_n\ ((\mathcal{T}_{T_1}\ x_1)\wedge\ldots\wedge(\mathcal{T}_{T_n}\ x_n))\Rightarrow(\varepsilon\ (\alpha_{T_n,o}\ \ldots\ (\alpha_{T_1,T_2\to\ldots\to T_n\to o}\ A\ x_1)\ \ldots\ x_n))\Leftrightarrow P$$

$$\exists f\ (\mathcal{T}_{T_1\to\ldots\to T_n\to U}\ f)\wedge\ \forall x_1\ \ldots\ \forall x_n\ ((\mathcal{T}_{T_1}\ x_1)\wedge\ldots\wedge(\mathcal{T}_{T_n}\ x_n))\Rightarrow(\alpha_{T_n,U}\ \ldots\ (\alpha_{T_1,T_2\to\ldots\to T_n\to U}\ f\ x_1)\ \ldots\ x_n))=_U t$$

Extensionality:

$$\forall P\ \forall Q\ ((\mathcal{T}_o\ P)\wedge(\mathcal{T}_o\ Q))\Rightarrow((\varepsilon\ P)\Leftrightarrow(\varepsilon\ Q))\Rightarrow(P=_o Q)$$

$$\forall f\ \forall g\ ((\mathcal{T}_{T\to U}\ f)\wedge(\mathcal{T}_{T\to U}\ g))\Rightarrow(\forall x\ (\mathcal{T}_T\ x)\Rightarrow(\alpha_{T,U}\ f\ x)=_U\ (\alpha_{T,U}\ g\ x))\Rightarrow(f=_{T\to U}\ g)$$

Equality:

$$\forall x\ (\mathcal{T}_T\ x)\Rightarrow(x=_T x)$$

$$\forall x_1\ \ldots\ \forall x_n\ \forall a\ \forall b\ ((\varepsilon(\mathcal{T}_{U_1}x_1))\wedge\ldots\wedge(\varepsilon(\mathcal{T}_{U_n}x_n))\wedge(\mathcal{T}_T\ a)\wedge(\mathcal{T}_T\ b))\Rightarrow(a=_T b)\Rightarrow(P[x\leftarrow a]\Rightarrow P[x\leftarrow b])$$

Typing:

$$\forall x\ \forall y\ ((\mathcal{T}_{T\to U}\ x)\wedge(\mathcal{T}_T\ y))\Rightarrow(\mathcal{T}_U\ (\alpha_{T,U}(x,y)))$$

Non empty:

$$\exists x\ (\mathcal{T}_\iota\ x)$$

Where $a=_T b$ is an abbreviation for $(\varepsilon\ (\alpha_{T,o}\ (\alpha_{T,T\to o}\ =_T\ a)\ b))$

Remark This translation is an extension of the one of [42] where a fragment of type theory without functions and with only unary sets is translated in an untyped setting.

Remark In the axioms schemes the schematic variables represent translations of propositions and terms of the typed language and not any proposition and term in the translated language.

Remark The axiom

$$\exists x\ (\mathcal{T}_T\ x)$$

is subsumed by the comprehension schemes and the non emptyness axiom for the type ι.

Remark We have seen that there are two ways to restrict naïve set theory to avoid Russell's paradox. Either we restrict the formation of propositions from objects (type theory) or we restrict the formation of objects from propositions (set theory). When we translate type theory into an untyped setting we shift from one restriction to the other, indeed the formation of proposition is not restricted anymore, but the relativization of quantifier restricts the comprehension schemes.

Relativization of quantifiers shows that, as already remarked in [42, 18], the two ways to avoid Russell's paradox: restricting the formation of propositions from objects (as in *type theory*) and restricting the formation of objects from propositions (as in *set theory*) lead to a somehow similar result.

3.2.2 A single symbol for application

From the theory above, we remove the symbols $\alpha_{T,U}$ and we add a symbol α. We then define the translation from a theory to the other, by replacing every $\alpha_{T,U}$ by α.

3.2.3 Adding equality

We add a symbol $=$ and we take as axioms the universal closure of the following propositions.

$$\forall x\ (x = x)$$

$$\forall a\ \forall b\ (a = b) \Rightarrow (P[x \leftarrow a] \Rightarrow P[x \leftarrow b])$$

$$\forall x\ \forall y\ (((\mathcal{T}_T\ x) \wedge (\mathcal{T}_T\ y)) \Rightarrow (x =_T y) \Leftrightarrow (x = y))$$

Remark The symbol $=_T$ is an individual symbol, while $=$ is a predicate symbol.

3.2.4 Extending predicates

If a is a predicate of type $T \rightarrow o$ and b an object that is not in T, then the proposition $(\varepsilon\ (\alpha\ a\ b))$ is not well-formed in the typed first order presentation of type theory. Now this proposition is well-formed, but we cannot prove it nor its negation. Is is convenient to add an axiom expressing that such a proposition is false, i.e. if a has type $T \rightarrow o$ then it does not contain objects out of T. In other words, atomic meaningless propositions are considered as false [42].

$$\forall a\ \forall b\ ((\mathcal{T}_{T \rightarrow o}\ a) \wedge (\varepsilon\ (\alpha\ a\ b))) \Rightarrow (\mathcal{T}_T\ b)$$

Remark Notice that when we say that an object is in $T \rightarrow U$ we say that when we apply it to an object in T, we get an object in U. With the axiom above we say a little more, if an object is in $T \rightarrow o$, when we apply it to an object in T we get an object in o, but also when we apply it to an object not in T we get an object which is not the positive truth value of o.

3.3 Types as sets

To the theory above, we add the individual symbols I, O and a function symbol $\rightarrow$ and the axiom

$$\forall a\ \forall b\ \forall f\ \forall x\ ((\varepsilon\ (\alpha\ (a \rightarrow b)\ f)) \wedge (\varepsilon\ (\alpha\ a\ x))) \Rightarrow (\varepsilon\ (\alpha\ b\ (f\ x)))$$

To each type T we associate a term T' defined by $\iota' = I$, $o' = O$, $(T \rightarrow U)' = T' \rightarrow U'$. From now on, a *type* is a term in the image of the translation $'$, i.e. a closed term in the sublanguage $I, O, \rightarrow$. We remove the symbols $\mathcal{T}_T$ and we replace every proposition $(\mathcal{T}_T\ a)$ by $(\varepsilon\ (\alpha\ T'\ a))$. We then get the following axioms.
Comprehension:

$$\exists A\ (\varepsilon\ (\alpha\ (T_1 \rightarrow ... \rightarrow T_n \rightarrow O)\ A)) \wedge \forall x_1\ ...\ \forall x_n\ (\varepsilon\ (\alpha\ T_1\ x_1)) \wedge ... \wedge (\varepsilon\ (\alpha\ T_n\ x_n)) \Rightarrow (\varepsilon\ (\alpha\ ...\ (\alpha\ A\ x_1)\ ...\ x_n)) \Leftrightarrow P$$

$$\exists f\ (\varepsilon\ (\alpha\ (T_1 \rightarrow ... \rightarrow T_n \rightarrow U)\ f)) \wedge \forall x_1\ ...\ \forall x_n\ ((\varepsilon\ (\alpha\ T_1\ x_1)) \wedge ... \wedge (\varepsilon\ (\alpha\ T_n\ x_n))) \Rightarrow (\alpha\ ...\ (\alpha\ f\ x_1)\ ...\ x_n)) = t$$

Extensionality:

$$\forall P\ \forall Q\ (\varepsilon\ (\alpha\ O\ P)) \wedge (\varepsilon\ (\alpha\ O\ Q)) \Rightarrow ((\varepsilon\ P) \Leftrightarrow (\varepsilon\ Q)) \Rightarrow (P = Q)$$

$$\forall f\ \forall g\ ((\varepsilon\ (\alpha\ (T \rightarrow U)\ f)) \wedge (\varepsilon\ (\alpha\ (T \rightarrow U)\ g))) \Rightarrow (\forall x\ (\varepsilon\ (\alpha\ T\ x)) \Rightarrow (\alpha\ f\ x) = (\alpha\ g\ x)) \Rightarrow (f = g)$$

Equality:

$$\forall x\ (x = x)$$

$$\forall a\ \forall b\ (a = b) \Rightarrow (P[x \leftarrow a] \Rightarrow P[x \leftarrow b])$$

$$\forall x\ \forall y\ ((\varepsilon\ (\alpha\ T\ x)) \wedge (\varepsilon\ (\alpha\ T\ y))) \Rightarrow (x =_T y) \Leftrightarrow (x = y)$$

Function space:

$$\forall a\ \forall b\ \forall f\ \forall x\ ((\varepsilon\ (\alpha\ (a \rightarrow b)\ f)) \wedge (\varepsilon\ (\alpha\ a\ x))) \Rightarrow (\varepsilon\ (\alpha\ b\ (f\ x)))$$

$$\forall a\ \forall b\ (((\varepsilon\ (\alpha\ (T \to O)\ a)) \wedge (\varepsilon\ (\alpha\ a\ b))) \Rightarrow (\varepsilon\ (\alpha\ T\ b)))$$

Non empty:

$$\exists x\ (\varepsilon\ (\alpha\ I\ x))$$

Remark The typing axiom

$$\forall x\ \forall y\ ((\varepsilon\ (\alpha\ (T \to U)\ x)) \wedge (\varepsilon\ (\alpha\ T\ y))) \Rightarrow (\varepsilon\ (\alpha\ U\ (\alpha\ x\ y)))$$

is subsumed by the function space axioms.

In a relaxed notation, we write $(a\ b)$ for the term $(\alpha\ a\ b)$. The axioms are rephrased as follows.
Comprehension:

$$\exists A\ (\varepsilon\ (T_1 \to ... \to T_n \to O)\ A) \wedge \forall x_1\ ...\ \forall x_n\ ((\varepsilon\ (T_1\ x_1)) \wedge ... \wedge (\varepsilon\ (T_n\ x_n) \Rightarrow (\varepsilon\ (A\ x_1\ ...\ x_n)) \Leftrightarrow P$$

$$\exists f\ (\varepsilon\ (T_1 \to ... \to T_n \to U)\ f) \wedge\ \forall x_1\ ...\ \forall x_n\ ((\varepsilon\ (T_1\ x_1)) \wedge ... \wedge (\varepsilon\ (T_n\ x_n))) \Rightarrow (f\ x_1\ ...\ x_n) = t$$

Extensionality:

$$\forall P\ \forall Q\ (\varepsilon\ (O\ P)) \wedge (\varepsilon\ (O\ Q)) \Rightarrow ((\varepsilon\ P) \Leftrightarrow (\varepsilon\ Q)) \Rightarrow (P = Q)$$

$$\forall f\ \forall g\ ((\varepsilon\ (T \to U)\ f) \wedge (\varepsilon\ (T \to U)\ g)) \Rightarrow (\forall x\ (\varepsilon\ (T\ x)) \Rightarrow (f\ x) = (g\ x)) \Rightarrow (f = g)$$

Equality:

$$\forall x\ (x = x)$$

$$\forall a\ \forall b\ (a = b) \Rightarrow (P[x \leftarrow a] \Rightarrow P[x \leftarrow b])$$

$$\forall x\ \forall y\ ((\varepsilon\ (T\ x)) \wedge (\varepsilon\ (T\ y))) \Rightarrow (x =_T y) \Leftrightarrow (x = y)$$

Function space:

$$\forall a\ \forall b\ \forall f\ \forall x\ ((\varepsilon\ ((a \to b)\ f)) \wedge (\varepsilon\ (a\ x))) \Rightarrow (\varepsilon\ (b\ (f\ x)))$$

$$\forall a\ \forall b\ (((\varepsilon\ ((T \to O)\ a)) \wedge (\varepsilon\ (a\ b))) \Rightarrow (\varepsilon\ (T\ b)))$$

Non empty:

$$\exists x\ (\varepsilon\ (I\ x))$$

Remark The comprehension schemes are indexed by propositions and terms. We have seen above how to avoid this dependence by using the combinators K and S. Many axioms are schemes indexed by types. This dependence can also be avoided by introducing a new predicate symbol $Type$ and axioms

$$Type(I)$$

$$Type(O)$$

$$\forall x\ \forall y\ (Type(x) \wedge Type(y)) \Rightarrow Type(x \to y)$$

and by replacing the type scheme variables by ordinary variables quantified in $Type$. For instance, the Second extensionality axiom would be rephrased

$$\forall T\ \forall U\ (Type(T) \wedge Type(U)) \Rightarrow \forall f\ \forall g\ ((\varepsilon\ (T \to U)\ f) \wedge (\varepsilon\ (T \to U)\ g)) \Rightarrow (\forall x\ (\varepsilon\ (T\ x)) \Rightarrow (f\ x) = (g\ x)) \Rightarrow (f = g)$$

This way, type theory can be formalized with a finite number of axioms.

3.4 Conservative extension

Definition Let P be a proposition in the language of type theory as a typed first order theory. We write $\Phi(P)$ for the proposition obtained by

- translating it into the untyped first order formulation,
- replacing the symbols $\alpha_{T,U}$ by α,
- replacing the propositions of the form $(\mathcal{T}_T\ a)$ by $(\varepsilon\ (\alpha\ T'\ a))$.

Proposition Let Γ be the axioms of type theory as a typed first order theory and Γ' the axioms of type theory above. Let Δ be a set of propositions in the typed language. The theory $\Gamma \cup \Delta$ has a model if and only if $\Gamma' \cup \Phi(\Delta)$ has a model.

Proof Let us consider a model $(M_T)_T$ of $\Gamma \cup \Delta$, without loss of generality, we can assume that this model is general, i.e. it is an equality functional model. Let us call y the unique element of Y.

We construct a model of $\Gamma' \cup \Phi(\Delta)$. We take $M = \biguplus M_T \uplus \{\bot\}$.

The denotation of the function symbol α is defined as follows. If a is in $M_{T\to U}$ and b is in M_T then $\overline{\alpha}(a,b) = a(b)$, otherwise $\overline{\alpha}(a,b) = \bot$. The denotation of the predicate symbol ε is the set $\{y\}$.

The denotation of the function symbol $\to$ is defined as follows. If a is in $M_{T\to o}$ and b is in $M_{U\to o}$ then consider the subset A of M_T of the x such that $a(x) = y$ and the subset M of M_U of the x such that $b(x) = y$. Then call C the subset of $M_{T\to U}$ of the f such that if $x \in A$ then $f(x) \in B$. Let c be the element of $M_{(T\to U)\to o}$ such that $c(x) = y$ if and only if $x \in C$, then we let $\overline{\Rightarrow}(a,b) = c$ (such an object exists, because the proposition $\exists c\ \forall f\ (c\ f) \Leftrightarrow (\forall x\ ((a\ x) \Rightarrow (b\ (f\ x))))$ is valid in the model $(M_T)_T$). Otherwise we let $\overline{\Rightarrow}(a,b) = \bot$.

The denotation of the symbol I is the element of $M_{\iota\to o}$ that maps every object of M_ι to y and the denotation of the symbol O is the element of $M_{o\to o}$ that maps every object of M_o to y (again, such objects exists because $(M_T)_T$ is a model of the comprehension schemes). The denotation of the symbol $=$ is the function mapping a and b to 1 if $a = b$ and to 0 otherwise. The denotation of the symbol $=_T$ is the function mapping a and b to y if $a = b$ and a and b are in M_T and to the other element of M_o otherwise (such an object exists, because $(M_T)_T$ is an equality model).

Now, the denotation of the proposition $\Phi(P)$ in this model is the denotation of P in the model $(M_T)_T$. Thus the propositions of $\Phi(\Delta)$ are valid in this model. So are the comprehension schemes and extensionality axioms. Then the equality axioms and function space axioms are also valid in this model, thus all the propositions of $\Gamma' \cup \Phi(\Delta)$ are valid.

Conversely, consider a model M of $\Gamma' \cup \Phi(\Delta)$, without loss of generality we can consider that this model is an equality model, i.e. that $=$ denotes equality.

We construct a model of $\Gamma \cup \Delta$. We define M_T as the subset of M of the elements a such that $\overline{\varepsilon}(\overline{\alpha}(\overline{T}, a)) = 1$.

The denotation of $\alpha_{T,U}$ is the restriction of the denotation of α to $M_{T\to U} \times M_T$, by the function space axioms the codomain of this function is in M_U. The denotation of ε is the restriction of the denotation of ε to M_o. The denotation of $=_T$ is the element of $M_{T\to T\to o}$ mapping a and b to y if $a = b$ and to the other element of M_o otherwise (such an object exists because the model is an equality model and a model of the proposition $\forall x\ \forall y\ ((\varepsilon\ (\alpha\ T\ x)) \wedge (\varepsilon\ (\alpha\ T\ y))) \Rightarrow (x =_T y) \Leftrightarrow (x = y)$).

Now, the denotation of a proposition P in this model is the denotation of $\Phi(P)$ in the model M. Thus the propositions of Δ are valid in this model. So are the comprehension schemes and extensionality axioms. Then the equality axioms are valid in this model, thus all the propositions of $\Gamma \cup \Delta$ are valid.

Corollary Let Γ be the axioms of type theory as a typed first order theory and Γ' of type theory above. Let Δ be a set of propositions and P be a proposition, we have $\Gamma\Delta \vdash P$ if and only if $\Gamma'\Phi(\Delta) \vdash \Phi(P)$.

Remark We can prove the consistency of type theory by building a model as follows. We take M_ι being any non empty set (infinite, if we consider an axiom of infinity), say the set of natural numbers, $M_o = \{0,1\}$ and $M_{T\to U}$ as the set of all functions from M_T to M_U. Then we interpret equality as equality in the model.

From this model we can build a model for the theory above by considering $M = \biguplus M_T \uplus \{\bot\}$.

Then we can wonder in which language this consistency proof can be formalized. To form the union (disjoint or not) of the collection of the M_T we need the union axiom and this collection to be a set. Thus, it seems that we require the replacement axiom to form such a set. Even with the replacement axiom, we need the existence of a functional relation $P(X, Y)$ which is true is and only if $X = T$ and $Y = M_T$ for some T. The possibility to construct this model of type theory in Zermelo's set theory or in Zermelo-Fraenkel set theory is not that obvious. We can prove the consistency of type theory in Zermelo's set theory, using the compactness theorem and the fact any finite set of propositions uses only a finite number of types. This way we can build a model for any finite set of propositions as a finite union of some $(M_T)_T$ (finite unions exists in Zermelo's set theory because we have the pairing axiom) [33, 30]. But, this method does not seem to generalize to the formulation of type theory where types are sets, as we cannot interpret the symbol $\to$ in a finite union of M_T's.

In contrast, the model above can be easily be built in Zermelo's set theory plus the axiom "there exists a model of Zermelo's set theory" as a subset of the model of set theory (relative consistency).

Remark It is well-know that cut-elimination for type theory is much more difficult than cut-elimination for first order logic. This is not contradictory with the possibility to express type theory as a first order theory. Indeed, the cut-elimination theorem for bare first order logic does not imply the cut elimination theorem for all the first order theories. For instance, in arithmetic we have more cuts than in bare first order logic. An open problem is to define a notion of cut for type theory as a first order language, that corresponds to the notion of cut in the usual formulation of type theory and to adapt the proofs of cut elimination [45, 41] and normalization [21, 22].

3.5 An extension

In the formulation above, in the comprehension schemes t and P need to be translations of propositions and terms of the typed presentation. We can extend these axioms in such a way that P is any proposition and t any term. However, we need to restrict the functional comprehension scheme.

Comprehension:

$$\exists A\ (\varepsilon\ (T_1 \to ... \to T_n \to O)\ A) \wedge \forall x_1\ ...\ \forall x_n\ ((\varepsilon\ (T_1\ x_1)) \wedge ... \wedge (\varepsilon\ (T_n\ x_n))) \Rightarrow ((\varepsilon\ (A\ x_1\ ...\ x_n)) \Leftrightarrow P))$$

$$\exists f\ (\varepsilon\ (T_1 \to ... \to T_n \to U)\ f) \wedge\ \forall x_1\ ...\ \forall x_n\ (((\varepsilon\ (T_1\ x_1)) \wedge ... \wedge (\varepsilon\ (T_n\ x_n)) \wedge (\varepsilon\ (U\ t))) \Rightarrow (f\ x_1\ ...\ x_n) = t)$$

This theory is consistent, as one can build a model for it from a standard model of type theory, i.e. a general model such that the set $M_{T \to U}$ is the set of all functions from M_T to M_U. But we leave open the problem of determining if it is an extension of the theory above, or if theorems are the same in both theories.

3.6 An explicit notation for sets and functions

Now consider the skolemization of the comprehension schemes. For each proposition P, sequence of variables $x_1, ..., x_n$ and sequence of types $T_1, ..., T_n$ we introduce an object $\{(x_1, T_1), ..., (x_n, T_n) \mid P\}$ and for each term t sequences of variables $x_1, ..., x_n$ and types $T_1, ..., T_n$ and type U we introduce an object $(x_1, T_1), ..., (x_n, T_n), U \mapsto t$. The comprehension schemes are rephrased

$$(\varepsilon\ ((T_1 \to ... \to T_n \to O)\ \{(x_1, T_1), ..., (x_n, T_n) \mid P\}))$$

$$\forall x_1\ ...\ \forall x_n\ ((\varepsilon\ (T_1\ x_1) \wedge ... \wedge (\varepsilon\ (T_n\ x_n) \Rightarrow (\varepsilon\ (\{(x_1, T_1), ..., (x_n, T_n) \mid P\}\ x_1\ ...\ x_n)) \Leftrightarrow P$$

$$(\varepsilon\ ((T_1 \to ... \to T_n \to U)\ ((x_1, T_1), ..., (x_n, T_n), U \mapsto t)))$$

$$\forall x_1\ ...\ \forall x_n\ ((\varepsilon\ (T_1\ x_1)) \wedge ... \wedge (\varepsilon\ (T_n\ x_n)) \wedge (\varepsilon\ (U\ t))) \Rightarrow (((x_1, T_1), ..., (x_n, T_n), U \mapsto t)\ x_1\ ...\ x_n) = t$$

In this presentation we have an explicit notation for objects, but this notation is closer to a language of combinators than to λ-calculus [14].

Remark In this language, we can drop the symbols $=_T$ because we have $\{(x_1,T),(x_2,T) \mid x_1 = x_2\}$.

Remark (Decidability of typing propositions)
Consider a closed term a in this language, there exists at most one type T such that $(\varepsilon\ (T\ a))$ is provable and this type can be computed from a. This also holds if a has free variables, provided that we have a unique axiom $(\varepsilon\ (T_x\ x))$ for every variable x free in a.

Remark (Turning the skolemized comprehension schemes into a rewrite rule)
In Church's type theory, β-equivalent propositions are provably equivalent. Thus we can identify β-equivalent propositions and drop the conversion axioms. This way, we erase the conversion steps from proofs. As β-equivalence is decidable, proof judgements are still decidable.
We want to do the same thing in this presentation of type theory. Let $a_1, ..., a_n$ be terms, from the axiom

$$\forall x_1\ ...\ \forall x_n\ ((\varepsilon\ (T_1\ x_1)) \wedge ... \wedge (\varepsilon\ (T_n\ x_n)) \wedge (\varepsilon\ (U\ t))) \Rightarrow (((x_1,T_1),...,(x_n,T_n),U \mapsto t)\ x_1\ ...\ x_n) = t$$

we get

$$((\varepsilon\ (T_1\ a_1)) \wedge ... \wedge (\varepsilon\ (T_n\ a_n)) \wedge (\varepsilon\ (U\ t[x_1 \leftarrow a_1,...,x_n \leftarrow a_n]))) \\ \Rightarrow (((x_1,T_1),...,(x_n,T_n),U \mapsto t)\ a_1\ ...\ a_n) = t[x_1 \leftarrow a_1,...,x_n \leftarrow a_n]$$

Thus, in this presentation, the conversion schemes (skolemized comprehension schemes) is guarded by typing conditions. As these predicates are decidable, β-equivalence is still decidable, but reduction requires dynamic type checking, i.e. type checking of expressions before each reduction step. This dynamic type checking is a drawback of too liberal languages where the formation of propositions is not restricted.

Remark We have seen that to restrict naïve set theory, we can either restrict the formation of propositions from objects (using typed languages) or the formation of objects from propositions (by restricting the comprehension schemes). Here we do not restrict the syntax of propositions neither the one of objects, but we restrict the *uses* of objects (i.e. the skolemized comprehension schemes) as to reduce $((x,T),U \mapsto t)\ u$ to $t[x \leftarrow u]$ we need to prove that u belongs to T and $t[x \leftarrow u]$ belongs to U.

3.7 Relations between type theory and set theory

3.7.1 A weak set theory

Extensionality axiom From the extensionality axioms we get

$$\forall a\ \forall b\ ((\varepsilon\ (T \to O)\ a) \wedge (\varepsilon\ (T \to O)\ b)) \Rightarrow (\forall x\ (\varepsilon\ (T\ x)) \Rightarrow ((\varepsilon\ (a\ x)) \Leftrightarrow (\varepsilon\ (b\ x))) \Rightarrow (a = b))$$

From which we can deduce

$$\forall a\ \forall b\ ((\varepsilon\ (T \to O)\ a) \wedge (\varepsilon\ (T \to O)\ b)) \Rightarrow (\forall x\ ((\varepsilon\ (a\ x)) \Leftrightarrow (\varepsilon\ (b\ x))) \Rightarrow (a = b))$$

If we write $(\varepsilon\ (x\ y))$ as $y \in x$ then the theorem above is rewritten

$$\forall a\ \forall b\ ((a \in (T \to O)) \wedge (b \in (T \to O))) \Rightarrow (\forall x\ ((x \in a) \Leftrightarrow (x \in b))) \Rightarrow (a = b)$$

This extensionality axiom is weaker than the usual one in set theory. As to prove the equality of two objects we must prove that they have the same elements, but also that they are sets and sets in the same type.
Thus this axiom allows to have non-sets elements (the so-called *urelements*), as the elements of I for instance and it allows several empty sets (one in each type). A similar restriction of the extensionality axiom is considered in [38]. However, the axiom of [38] allows non-sets elements but only one empty set.

Subset scheme Consider a proposition P and call x, $y_1, ..., y_n$ its free variables. Let a be a variable. By the comprehension scheme we have

$$\exists b\ ((\varepsilon\ ((U_1 \to ... \to U_n \to (T \to O) \to T \to O)\ b)) \wedge \forall y_1\ ...\ \forall y_n\ \forall a\ \forall x$$
$$(((\varepsilon\ (U_1\ y_1)) \wedge ... \wedge (\varepsilon\ (U_n\ y_n)) \wedge (\varepsilon\ ((T \to O)\ a)) \wedge (\varepsilon\ (T\ x))) \Rightarrow ((\varepsilon\ (b\ y_1\ ...\ y_n\ a\ x)) \Leftrightarrow ((\varepsilon\ (a\ x)) \wedge P))))$$

From which we can derive

$$\forall y_1\ ...\ \forall y_n\ \forall a\ (((\varepsilon\ (U_1\ y_1)) \wedge ... \wedge (\varepsilon\ (U_n\ y_n)) \wedge (\varepsilon\ ((T \to O)\ a))) \Rightarrow \exists c\ \forall x\ ((\varepsilon\ (c\ x)) \Leftrightarrow (\varepsilon\ (a\ x)) \wedge P)))$$

If we write $(\varepsilon\ (x\ y))$ as $y \in x$ then the theorem above is rewritten

$$\forall y_1\ ...\ \forall y_n\ \forall a\ ((y_1 \in U_1) \wedge ... \wedge (y_n \in U_n) \wedge (a \in (T \to O))) \Rightarrow \exists c\ \forall x\ ((x \in c) \Leftrightarrow ((x \in a) \wedge P))$$

which is the subset scheme of set theory, the only difference is that we have a guard $a \in (T \to O)$ that expresses that a needs to be a set and the guards $y_i \in U_i$ that express that y_i must belong to some set.

Remark The set a carries the type T of the elements of b. Indeed, the comprehension scheme of type theory gives the equivalence $(\varepsilon\ (b\ y_1\ ...\ y_n\ a\ x)) \Leftrightarrow ((\varepsilon\ (a\ x)) \wedge P)$ only for x in T. We can drop this hypothesis because if x is not in T then both $(\varepsilon\ (b\ y_1\ ...\ y_n\ a\ x))$ and $(a\ x)$ are false. In absence of a set a, we have to keep this hypothesis and we get the equivalence $(\varepsilon\ (b\ y_1\ ...\ y_n\ x)) \Leftrightarrow ((\varepsilon\ (T\ x)) \wedge P)$.

Power set axiom By the comprehension scheme we have

$$\exists b\ (\varepsilon\ ((T \to O) \to O)\ b) \wedge \forall x\ (\varepsilon\ ((T \to O)\ x)) \Rightarrow (\varepsilon\ (b\ x)) \Leftrightarrow (\forall y\ (\varepsilon\ (T\ y)) \Rightarrow (\varepsilon\ (x\ y) \Rightarrow (\varepsilon\ (a\ y))))$$

As above we deduce

$$\forall a\ (\varepsilon\ ((T \to O)\ a)) \Rightarrow \exists b\ \forall x\ (\varepsilon\ (b\ x)) \Leftrightarrow ((\varepsilon\ ((T \to O)\ x)) \wedge (\forall y\ (\varepsilon\ (x\ y) \Rightarrow (\varepsilon\ (a\ y)))))$$

Again, if we write $(\varepsilon\ (x\ y))$ as $y \in x$ we get

$$\forall a\ (a \in (T \to O)) \Rightarrow \exists b\ \forall x\ (x \in b) \Leftrightarrow ((x \in (T \to O)) \wedge (\forall y\ (y \in x) \Rightarrow (y \in a)))$$

Which is the power set axiom. We have the same guard as above. Moreover we have an extra condition $x \in (T \to O)$. Notice that the theorem is false without this condition because the power set of a does not contain the empty sets of types different from $T \to O$.

Union axiom By the comprehension scheme we have

$$\exists b\ (\varepsilon\ ((T \to O)\ b)) \wedge \forall x\ (\varepsilon\ (T\ x)) \Rightarrow (\varepsilon\ (b\ x)) \Leftrightarrow \exists y\ ((\varepsilon\ ((T \to O)\ y)) \wedge (\varepsilon\ (y\ x)) \wedge (\varepsilon\ (a\ y)))$$

As above we deduce

$$\forall a\ (\varepsilon\ (((T \to O) \to O))\ a) \Rightarrow \exists b\ \forall x\ (\varepsilon\ (b\ x)) \Leftrightarrow \exists y\ (\varepsilon\ (y\ x)) \wedge (\varepsilon\ (a\ y)))$$

Again, if we write $(\varepsilon\ (x\ y))$ as $y \in x$ we get

$$\forall a\ (a \in ((T \to O) \to O)) \Rightarrow \exists b\ \forall x\ (x \in b) \Leftrightarrow \exists y\ ((x \in y) \wedge (y \in a))$$

Which is the union axiom with the same guard as above.

Pairing axiom By the comprehension scheme we have

$$\exists b\ (\varepsilon\ ((T \to O)\ b)) \wedge (\forall x\ (\varepsilon\ (T\ x)) \Rightarrow ((\varepsilon\ (b\ x)) \Leftrightarrow ((x = a) \wedge (x = a'))))$$

As above we deduce

$$\forall a\ \forall a'\ ((\varepsilon\ (T\ a)) \wedge (\varepsilon\ (T\ a'))) \Rightarrow \exists b\ (\forall x\ (\varepsilon\ (b\ x)) \Leftrightarrow ((x = a) \wedge (x = a')))$$

Again, if we write $(\varepsilon\ (x\ y))$ as $y \in x$ we get

$$\forall a\ \forall a'\ ((a \in T) \wedge (a' \in T)) \Rightarrow \exists b\ (\forall x\ (x \in b) \Leftrightarrow ((x = a) \wedge (x = a')))$$

Here the guard is very strong, to form the pair of two elements we need them to be in the same type. It is fairer to say that the pairing axiom is false this theory.

To allow a more liberal pairing axiom, we would need to include type cumulativity [11] and we would get a theory very close to set theory [42, 23].

3.7.2 A comparison

Both type theory and set theory, consider a universe that contains a set of primitive elements (I in type theory and $\{\emptyset\}$ in set theory) and is closed by two operations: taking the power set of some set and taking a subset of some set defined by a property expressed in the language. In set theory, these operations are expressed by the power set axiom and the subset scheme. Type theory exploits the fact that the power set of a subset of A is a subset of the power set of A and thus all the power set operations can be performed before all the subset operations. The power set operation is expressed by the types construction rules and the subset operation by the comprehension scheme. To both theory we can add an axiom of infinity, expressing that there is an infinite set.

In this universe the union axiom is admissible, i.e. by induction over the structure of the construction of A, if A is a set of sets, all the elements of the elements of A belong to some set B, thus the union of the elements of A can be build as a subset of B.

Then, these theories must give tools for relations (i.e. n-ary predicates). In type theory, we extend the comprehensions scheme to n-ary predicates. In set theory we add an axiom allowing the formation of pairs. Then ordered pairs are formed by the Haussdorf-Wiener-Kuratowski definition $< a, b >= \{\{a\}, \{a, b\}\}$. A n-uple $< a_1, a_2, ..., a_{n-1}, a_n >$ is then defined as $< a_1, < a_2, ..., < a_{n-1}, a_n > ... >>$ and a n-ary relation is a predicate over n-uples.

Then we need to form functions. In set theory functions are relations. In type theory, we extend the comprehension scheme for functional types and we add a descriptions axiom to relate functional relations and functions [6].

At last, we need to form natural numbers, none of the theories can define n as the set of sets of n elements. Both theory can use the infinite set to construct finite cardinals as the set of sets of n elements of this set (see for instance [6] for an example in type theory and [1] for one in set theory). But, using the pairing axiom, set theory can also give Von Neumann's definition of ordinals, i.e. take for the number n a the set of n elements $\{0, ..., n-1\}$.

3.7.3 Pairing in set theory

The pairing axiom in set theory is then used to build ordered pairs (and thus relations), finite unions and ordinals. But the cost is that this axiom permits to form heterogeneous sets such as $\{0, \{0\}\}$. Notice that this axiom is the only one responsible for heterogeneous sets in set theory: if we drop it we can associate a type to all the objects that we can form in set theory. Dropping this axiom is advocated, for instance, in [9]. Taking n-ary predicates and functions as primitive and defining natural numbers as cardinals permits to drop this axiom. In some cases, we might want to take cartesian products and disjoint unions as primitive constructions too (cartesian products and disjoint unions are built using the pairing axiom in set theory). But we leave open the problem of giving a definition for transfinite numbers in this setting.

4 An extension of type theory with subtyping

We consider, at last, an extension of type theory, where the axioms are not restricted by types, but by any set. The axioms are rephrased as follows.
Comprehension:

$$\forall a_1 \ldots \forall a_n \ \forall a'_1 \ldots \forall a'_n \ ((\varepsilon \ ((a'_1 \to O) \ a_1)) \wedge \ldots \wedge (\varepsilon \ ((a'_n \to O) \ a_n)))$$
$$\Rightarrow \exists A \ (\varepsilon \ (a_1 \to \ldots \to a_n \to O) \ A) \wedge \forall x_1 \ldots \forall x_n \ ((\varepsilon \ (a_1 \ x_1)) \wedge \ldots \wedge (\varepsilon \ (a_n \ x_n))) \Rightarrow ((\varepsilon \ (A \ x_1 \ \ldots \ x_n)) \Leftrightarrow P))$$

$$\forall a_1 \ldots \forall a_n \ \forall b \ \forall a'_1 \ldots \forall a'_n \ \forall b' \ \forall c \ ((\varepsilon \ ((a'_1 \to O) \ a_1)) \wedge \ldots \wedge (\varepsilon \ ((a'_n \to O) \ a_n)) \wedge (\varepsilon \ ((b' \to O) \ b)) \wedge (\varepsilon \ (b \ c)))$$
$$\Rightarrow \exists f \ (\varepsilon \ (a_1 \to \ldots \to a_n \to b) \ f) \wedge \forall x_1 \ldots \forall x_n \ (((\varepsilon \ (a_1 \ x_1)) \wedge \ldots \wedge (\varepsilon \ (a_n \ x_n)) \wedge (\varepsilon \ (b \ t))) \Rightarrow (f \ x_1 \ \ldots \ x_n) = t)$$

Extensionality:

$$\forall P \ \forall Q \ (\varepsilon \ (O \ P)) \wedge (\varepsilon \ (O \ Q)) \Rightarrow ((\varepsilon \ P) \Leftrightarrow (\varepsilon \ Q)) \Rightarrow (P = Q)$$

$$\forall a \ \forall b \ \forall f \ \forall g \ ((\varepsilon \ ((a \to b) \ f)) \wedge (\varepsilon \ ((a \to b) \ g))) \Rightarrow (\forall x \ (\varepsilon \ (a \ x)) \Rightarrow (f \ x) = (g \ x)) \Rightarrow (f = g)$$

Equality:

$$\forall x \ (x = x)$$

$$\forall a \ \forall b \ (a = b) \Rightarrow (P[x \leftarrow a] \Rightarrow P[x \leftarrow b])$$

Function space:

$$\forall a \ \forall b \ \forall f \ \forall x \ ((\varepsilon \ ((a \to b) \ f)) \wedge (\varepsilon \ (a \ x))) \Rightarrow (\varepsilon \ (b \ (f \ x)))$$

$$\forall a \ \forall b \ \forall c \ (((\varepsilon \ ((c \to O) \ a)) \wedge (\varepsilon \ (a \ b))) \Rightarrow (\varepsilon \ (c \ b)))$$

Non empty:

$$\exists x \ (\varepsilon \ (I \ x))$$

Proposition This theory is consistent.
Proof We construct a model as follows.

Take M_I be any non empty set (infinite if we consider also an axiom of infinity). Take $M_O = \{0, 1\}$. Take $M_{T \to U}$ be the set of functions from M_T to $(M_U \uplus \{\bot_U\})$. We define the set M as the set $\biguplus M_T \uplus \{\bot\}$.

The denotation of the function symbol α is defined as follows. If a is in $M_{T \to U}$ and b is in M_T and $a(b) \neq \bot_U$ then $\overline{\alpha}(a, b) = a(b)$, otherwise $\overline{\alpha}(a, b) = \bot$. The denotation of the predicate symbol ε is the set $\{1\}$.

The denotation of the function symbol $\to$ is defined as follows. If a is in $M_{T \to o}$ and b is in $M_{U \to o}$ then consider the subset A of M_T of the x such that $a(x) = 1$ and the subset B of M_U of the x such that $b(x) = 1$. Then call C the subset of $M_{T \to U}$ of the f such that if $x \in A$ then $f(x) \in B$ and $f(x) = \bot_U$ otherwise. Let c be the element of $M_{(T \to U) \to o}$ such that $c(x) = 1$ if and only if $x \in C$, then we let $\overline{\Rightarrow}(a, b) = c$. Otherwise we let $\overline{\Rightarrow}(a, b) = \bot$.

The denotation of the symbol I is the element of $M_{I \to O}$ that maps every object of M_I to 1. The denotation of the symbol O is the element of $M_{O \to O}$ that maps every object of M_O to 1. The denotation of the symbol $=$ is the equality on the domain.

We first prove that if the proposition $(\varepsilon \ ((b \to O) \ a))$ is valid, then $\overline{a}$ and $\overline{b}$ are in some $M_{T \to O}$ and $\overline{a}(x) \in O$ if $\overline{b}(x) = 1$ and $\overline{a}(x) = \bot_O$ otherwise.

We then check now that the axioms are valid in this model.

- Comprehension. If the denotation of some a_i is not in some $M_{T_i \to O}$ then $(\varepsilon \ ((a'_i \to O) \ a_i))$ is false. Otherwise, we take for A the function mapping $e_1, \ldots, e_n$ to 1 if for every i, $\overline{a_i}(e_i) = 1$ and P is verified by $e_1, \ldots, e_n$, to 0 if for every i, $\overline{a_i}(e_i) = 1$ and P is not verified by $e_1, \ldots, e_n$ and to $\bot_O$ otherwise. This function is an element of $a_1 \to \ldots \to a_n \to O$.

  In the same way, either the denotation of some a_i is not in some $M_{T_i \to O}$, the denotation of b is not in some $M_{U \to O}$ or the denotation of b in an "empty set", i.e. a function in $M_{U \to O}$ never taking the value 1 and the condition is always false, or f can be built as an element of $a_1 \to \ldots \to a_n \to b$.

- Extensionality. The first extensionality axiom is obviously valid. For the second, either a is not in some $M_{T\to O}$ or b is not in some $M_{U\to O}$ then the conditions are false. Otherwise the two functions coïncide on a and they are both equal to $\bot_U$ out of a. Thus they are equal.
- The equality axioms, the function space axioms and the non emptyness axiom are obviously valid.

Remark Again, the formation of this model is not obvious in Zermelo's of Zermelo-Fraenkel set theory, but it is simple in the set theory extended with an axiom "set theory is consistent".

Remark This formulation of type theory is typeless, as every set is a type.

Remark Even if *even* is a subset of *nat* the empty set of even numbers and the empty set of natural numbers are distinct. The proposition $\neg(\emptyset_{nat} = \emptyset_{even})$ is valid in the model above. Both $(\emptyset_{nat}\ 3)$ and $(\emptyset_{even}\ 3)$ are false, but the first is merely false and the second is false because it is meaningless.

In a syntactically restricted language, only the first proposition would be allowed.

Remark The language obtained by skolemizing these axiom, we have a term $f = (x, nat), even \mapsto 2 * x$, where even is the predicate $\{(n, nat) \mid \exists p\ (\varepsilon\ (nat\ p)) \wedge (n = 2 * p)\}$. This language handles subtyping as we can define the term $g = (x, even), even \mapsto f(x)$. The function f and g are different, for instance in the model above the proposition $\neg(f = g)$ is valid. In this setting, the types ordering is mere inclusion.

Ordinary subtyping can be recovered as some subset of this language, for instance to a limited collection of sets, where inclusion is decidable.

Remark The usual methodology in typed λ-calculus to introduce a new concept is to define a language for types and terms and then a model of this language in ordinary mathematics (set theory). It seems that we use a different method here by giving a definition in ordinary mathematics of the new concept. In fact it is well known that the two methods are almost the same. For instance, it is the same to define a group as a model of group theory or as a set with an internal binary function, that is associative, has a neutral element and such that every element has an inverse.

Conclusion

We have given a new formulation of type theory. Along the way we have given a modular proof of Henkin's completeness theorem, by reducing it to Gödel's completeness theorem for first order logic and a new proof of Miller's theorem for a combinators based type theory, by reducing it to Skolem's theorem for first order logic.

The formulation of type theory we have obtained is a first order theory. Types are mere sets, thus it eliminates one of the conceptual difficulties of type theory: explaining both the notion of type and the notion of set. In the formulation of the theory with a predicates for types, typical ambiguity is not required: group theory, for instance, can be developed in an arbitrary set of an arbitrary type.

The skolemized form of this theory gives a language for sets and functions based on combinators. Functions are primitive and propositions are objects.

Meaningless statements are not yet forbidden in this formulation (thus this formulation may be not as good as the usual one for automated theorem proving [6], and reduction on this language requires dynamic type checking). If we want to forbid such statements, while keeping types as sets, we can use neither first order logic nor typed first order logic. We need to develop a variant of first order logic where the well-formedness and truth of propositions are mutually dependent. In such a language, we need for instance, to prove that n is a natural number to form the proposition *even(n)*). Such a language, like natural languages, would provide three different notions: syntactical correctness, meaningfulness and truth, while first order languages (untyped or typed) provide only syntactical correctness and truth. Such a language may be a good alternative to subtyping.

At last it should be possible to express in an extension of this theory, the Heyting-Kolmogorov interpretation of intuitionistic proofs as functional objects. Propositions would be represented by sets of proofs instead of types of proofs and the decidability of proof judgements would be a consequence of the decidability of a fragment of the language, instead of the decidability of typing judgements. Although the *truth* of propositions and the *truth* of proof judgements are two distinct notions [35], propositions and proof judgements can be expressed in a single universal language, this language having two distinct semantics.

Acknowledgements

The author thanks Gérard Huet for many comments on a draft of this paper.

References

[1] J.R. Abrial, The B-book: assigning programs to meanings, Cambridge University Press, (to be published).

[2] P.B. Andrews, A reduction of the axioms of the theory of propositional types, *Fund. Math.* 52 (1963) pp. 345-350.

[3] P.B. Andrews, Resolution in type theory, *The Journal of Symbolic Logic*, 36, 3 (1971) pp. 414-432.

[4] P.B. Andrews, General models, descriptions and choice in type theory, *The Journal of Symbolic Logic*, 37, 2 (1972) pp. 385-394.

[5] P.B. Andrews, General models and extensionality, *The Journal of Symbolic Logic*, 37, 2 (1972) pp. 395-397.

[6] P.B. Andrews, An introduction to mathematical logic and type theory: to truth through proof, *Academic Press*, Orlando (1986).

[7] J. Van Bethem and K. Doets, Higher order logic, *Handbook of Philosophical Logic*, D. Gabbay and F. Guenthner (eds.) I, *Reidel* (1983) pp. 275-329.

[8] N.G. de Bruijn, A survey of the project Automath, *To H.B. Curry : Essays on Combinatory Logic, Lambda Calculus and Formalism*, J.R. Hindley, J.P. Seldin (eds.), Academic Press (1980).

[9] Y. Caseau, Étude et réalisation d'un langage objet : LORE, *Thèse de Doctorat*, Université de Paris-Sud (1987).

[10] A. Church, A formulation of the simple theory of types, *The Journal of Symbolic Logic*, 5 (1940), pp. 56-68.

[11] L. Chwistek, The theory of constructive types, *Annales de la Société Polonaise de Mathématiques*, 2, (1924) pp. 9-48; 3 (1925) pp. 92-141.

[12] T. Coquand, G. Huet, The calculus of constructions, *Information and Computation*, 76 (1988) pp. 95-120.

[13] M. Davis, In defense of first order logic, *Talk*.

[14] G. Dowek, Lambda-calculus, combinators and the comprehension scheme, *Proceedings of Typed Lambda Calculi and Applications* (1995). Rapport de Recherche 2565, *Institut National de Recherche en Informatique et en Automatique* (1995).

[15] H.B. Enderton, A mathematical introduction to logic, *Academic Press*, New-York (1972).

[16] S. Feferman, A Language and axioms for explicit mathematics, *Algebra and Logic*, Lecture Notes in Mathematics 450, Springer-Verlag, Berlin (1975) pp. 87-139.

[17] S. Feferman, Theories of finite type related to mathematical practice, *Handbbok of Mathematical Logic*, J. Barwise (Ed.), North-Holland (1977), pp. 913-971.

[18] A. A. Fraenkel, Y. Bar-Hillel, A. Levy, Foundations of set theory, *North-Holland* (1973).

[19] H. Friedman, Equality between functionals, *Lecture Notes in Mathematics* 453, R. Parikh, (ed.), Springer-Verlag, Berlin (1975) pp. 23-37.

[20] J. Gallier, Logic in computer science, *Harper and Row*, New-York (1986).

[21] J.Y. Girard, Une extension de l'interprétation de Gödel à l'analyse et son application à l'élimination des coupures dans l'analyse et la théorie des types, *Proceedings of the* 2^{nd} *Scandinavian Logic Symposium*, Fenstad (ed.), *North-Holland*, Amsterdam (1970).

[22] J.Y. Girard, Interprétation fonctionnelle et élimination des coupures dans l'arithmétique d'ordre supérieur, *Thèse d'État*, Université de Paris 7 (1972).

[23] P. Gochet, P. Gribomont, Logique, *Hermès*, Paris (1990, 1994).

[24] L. Henkin, Completeness in the theory of types, *The Journal of Symbolic Logic*, 15 (1950) pp. 81-91.

[25] L. Henkin, A theory of propositional types, *Fund. Math.* 52 (1963) pp. 323-344.

[26] J. Herbrand, Recherches sur la théorie de la démonstration, Paris (1930).

[27] J. Hintikka (ed.), The philosophy of mathematics, *Oxford University Press*, Oxford (1969).

[28] G. Huet, A Mechanisation of Type Theory, *Proceedings of the Third International Joint Conference on Artificial Inteligence* (1973) pp. 139-146.

[29] G. Huet, A Unification Algorithm for Typed λ-calculus, *Theoretical Computer Science*, 1 (1975) pp. 27-57.

[30] J.G. Kemeny, Type theory vs. set theory, *The Journal of Symbolic Logic* 15, 1 (1950) p. 78.

[31] M. Kerber, How to prove higher order theorems in first order logic, *Twelfth International Joint Conference on Artificial Intelligence*, 1 (1991) pp. 137-142.

[32] J.L. Krivine, Lambda calculus, types and models, *Ellis Horwood* (1993).

[33] J. Lake, Comparing type theory and set theory, *Zeischrift für matematische Logik und Grundlagen der Mathematik*, 21 (1975) pp. 355-356.

[34] P. Martin-Löf, Intuitionistic type theory, *Bibliopolis*, Napoli (1984).

[35] P. Martin-Löf, Truth of a proposition, evidence of a judgement, validity of a proof, *Workshop on the theories of meaning*, Florence (1985).

[36] D.A. Miller, Proofs in higher order logic, *PhD Thesis*, Carnegie Mellon University (1983).

[37] D.A. Miller, A Compact representation of proofs, *Studia Logica*, 46, 4 (1987).

[38] R. Montague, Set theory and higher-order logic, *Formal systems and recursive functions*, J. Crossley and M. Dummett (eds.), *North-Holland* (1965).

[39] A. P. Morse, A theory of sets, *Academic-Press*, New-York (1965).

[40] L.C. Paulson, Set theory for verification, I. From foundations to functions, *Journal of Automated Reasoning*, 11 (1993) pp. 353-389.

[41] D. Prawitz, Hauptsatz for higher order logic, *The Journal of Symbolic Logic*, 33 (1968) pp. 452-457.

[42] W.V.O. Quine, Set theory and its logic, *Belknap press*, Cambridge Massachusets (1969).

[43] A. Schmidt, Über deductive theorien mit mehreren sorten von grunddigen, *Mathematische Annalen*, 115 (1938), pp. 485-506.

[44] D. Scott, Axiomatizing set theory, *Proceedings of symposia in pure mathematics*, 13, II (1974) pp. 207-214.

[45] M.O. Takahashi, A Proof of cut-elimination in simple type theory, *Journal of the Mathematical Society of Japan*, 19 (1967) pp. 399-410.

[46] H. Wang, Logic of many-sorted theories, *The Journal of Symbolic Logic*, 17, 2, (1952), pp. 105-116.

Unité de recherche Inria Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 Villers Lès Nancy
Unité de recherche Inria Rennes, Irisa, Campus universitaire de Beaulieu, 35042 Rennes Cedex
Unité de recherche Inria Rhône-Alpes, 46 avenue Félix Viallet, 38031 Grenoble Cedex 1
Unité de recherche Inria Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 Le Chesnay Cedex
Unité de recherche Inria Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 Sophia-Antipolis Cedex

Éditeur
Inria, Domaine de Voluceau, Rocquencourt, BP 105, 78153 Le Chesnay Cedex (France)
ISSN 0249-6399